

TUGAS FINAL
Pemrograman Mobile
Membuat Aplikasi ToDo



Di Susun Oleh :

JABALNUR

D121181313

Link

<https://github.com/JabalnurIT/ToDo-Final.git>

Departemen Teknik Informatika

Fakultas Teknik

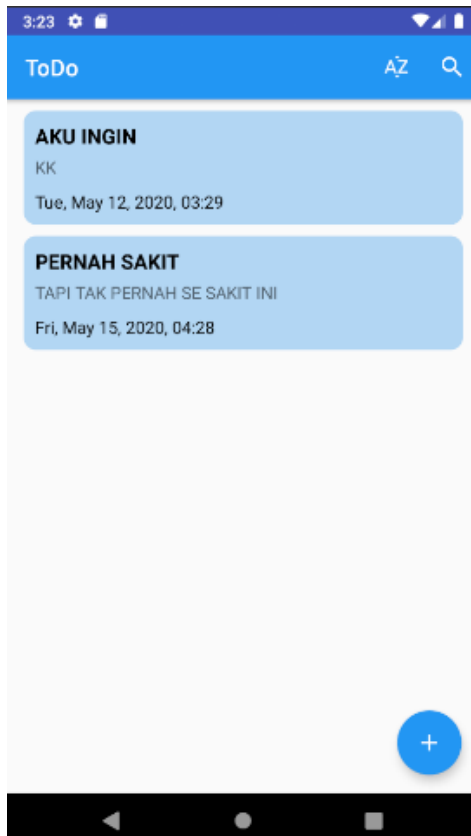
Universitas Hasanuddin

2020

Link

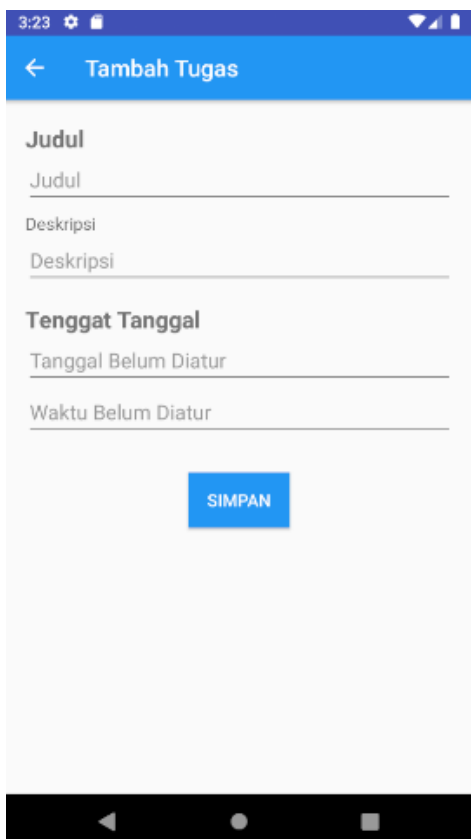
<https://github.com/JabalnurIT/ToDo-Final.git>

Penjelasan Dari Aplikasi



Disamping adalah tampilan awal dari aplikasi ToDo kita.

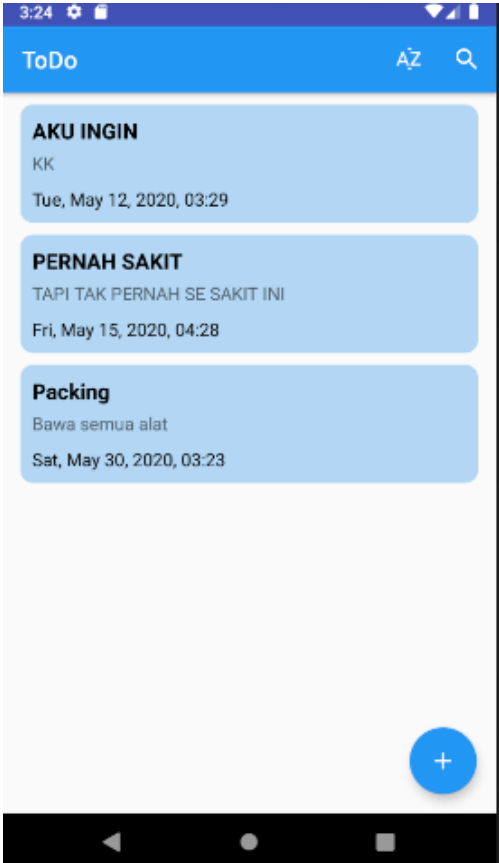
Tampilannya memiliki tombol tambah untuk menambah tugas, tombol sort untuk mengurutkan tugas, dan tombol search untuk mencari tugas



Saat kita menekan tombol tambah tugas, tampilan akan berubah menjadi seperti ini. Kita diminta untuk memasukkan Judul tugas, deskripsi dan Tenggat. Dan Kolom tersebut tidak bisa kosong

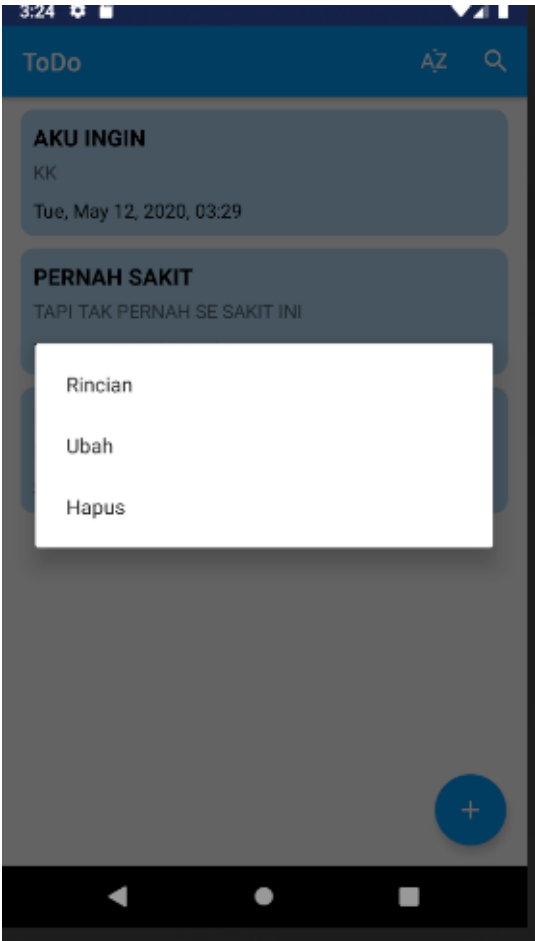


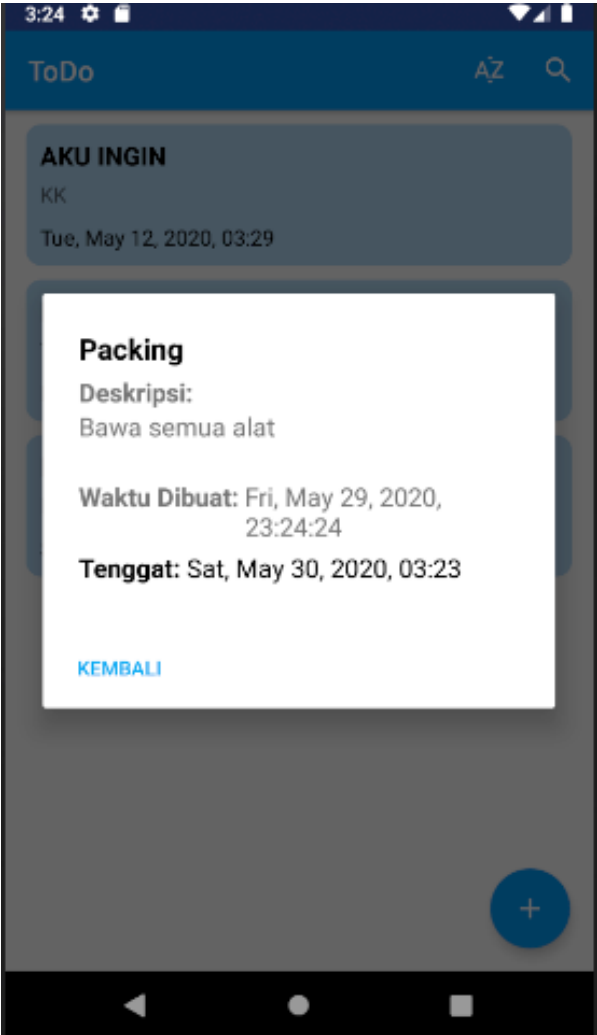
Apabila kita menekan tombol simpan dan terdapat kolom yang kosong, maka akan muncul Toasat yang mengatakan tidak boleh kosong



Setelah kita menambahkan tugas baru dan menekan simpan, makan tampilan aplikasi akan kembali ke tampilan awal dan tugas tersebut akan muncul.

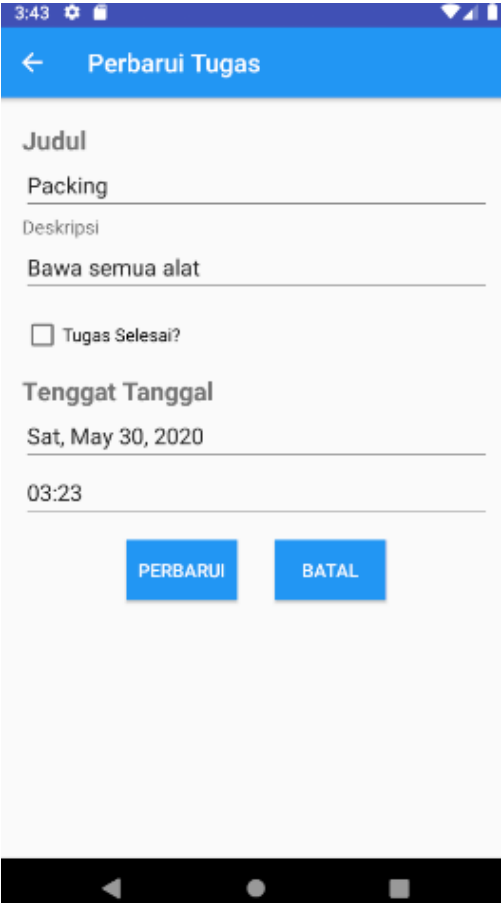
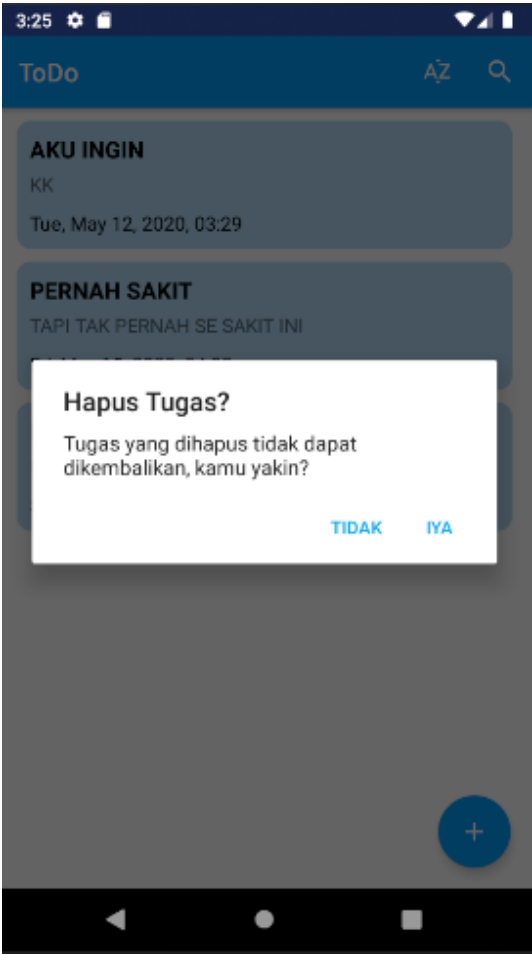
Jika kita menekan salah satu tombol tugas, maka akan muncul menu dengan tiga pilihan



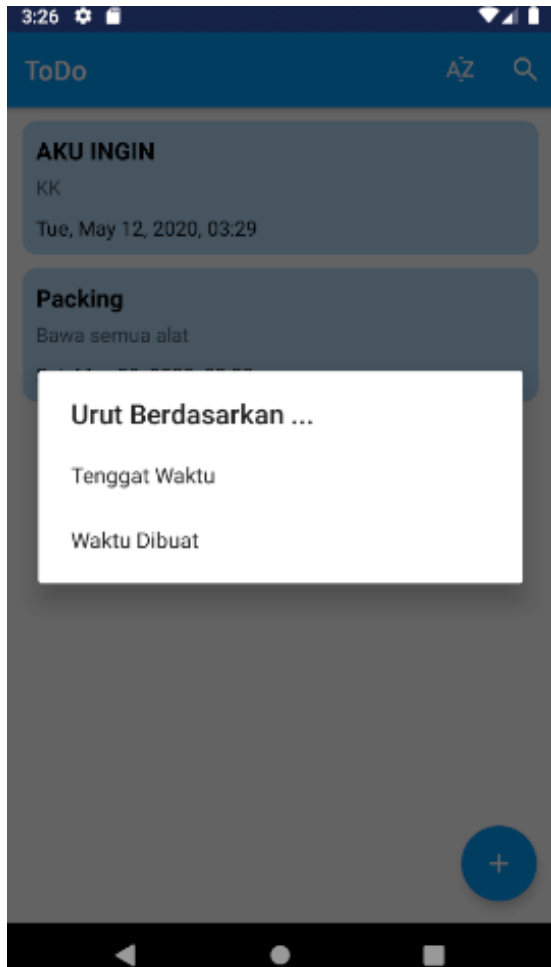


Jika Kita menekan Rincian, maka akan muncul rincian dari tugas tersebut. Kita hanya tinggal menekan tombol kembali untuk keluar dari tampilan itu.

Apabila kita menekan tombol hapus maka akan muncul peringatan



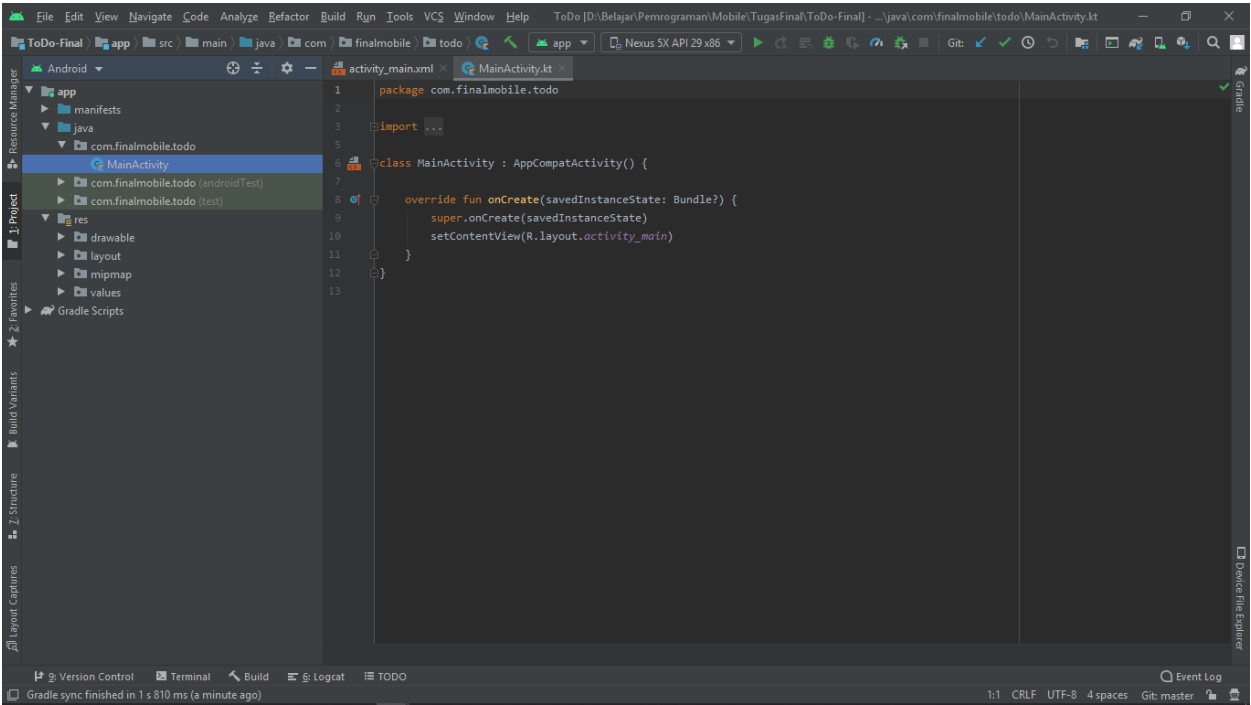
Jika Kita memilih Ubah, maka tampilannya akan berubah menjadi seperti ini. Disini, selain mengubah Judul, Deskripsi dan Tenggat, kita juga dapat menandai apakah tugas tersebut sudah selesai atau belum. Jika kita menandainya sebagai selesai, maka pada saat kita perbarui, tugasnya akan hilang



Jika kita menekan tombol sort dengan logo AZ, maka akan muncul menu untuk mengurutkan berdasarkan Tenggat, dan Waktu pembuatan

DOKUMENTASI PEMBUATAN

Commit Awal



Pada bagian ini kita hanya membuat project awal kita dengan memilih menu File->New->New Project

Commit Gradel

Pada bagain ini kita akan memasukkan gradle yang akan kita gunakan, pastikan juga menggunakan versi yang terbaru.

```
build.gradle (app) x build.gradle (ToDo) x
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    ext.kotlin_version = '1.3.72'
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.6.3'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

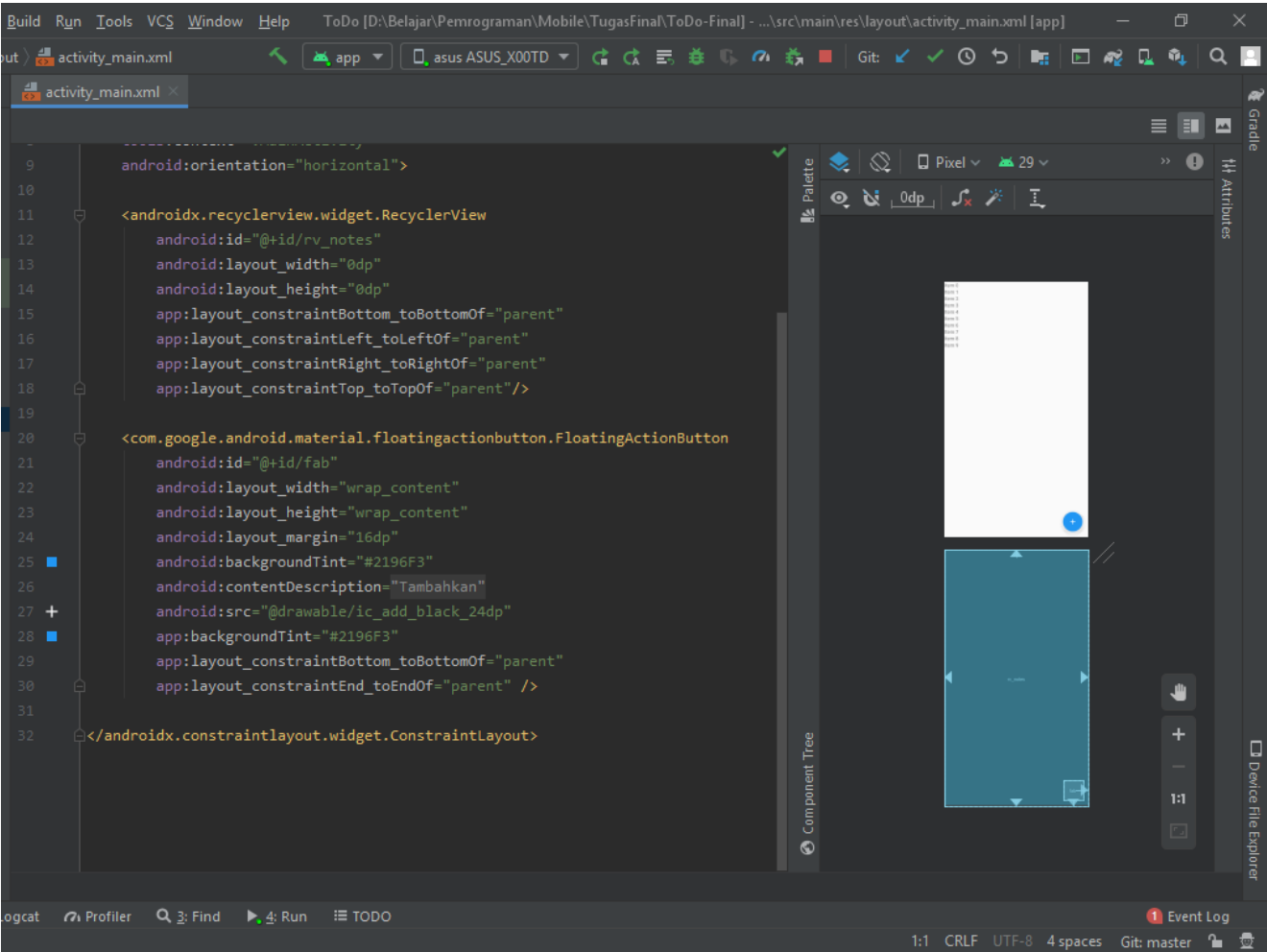
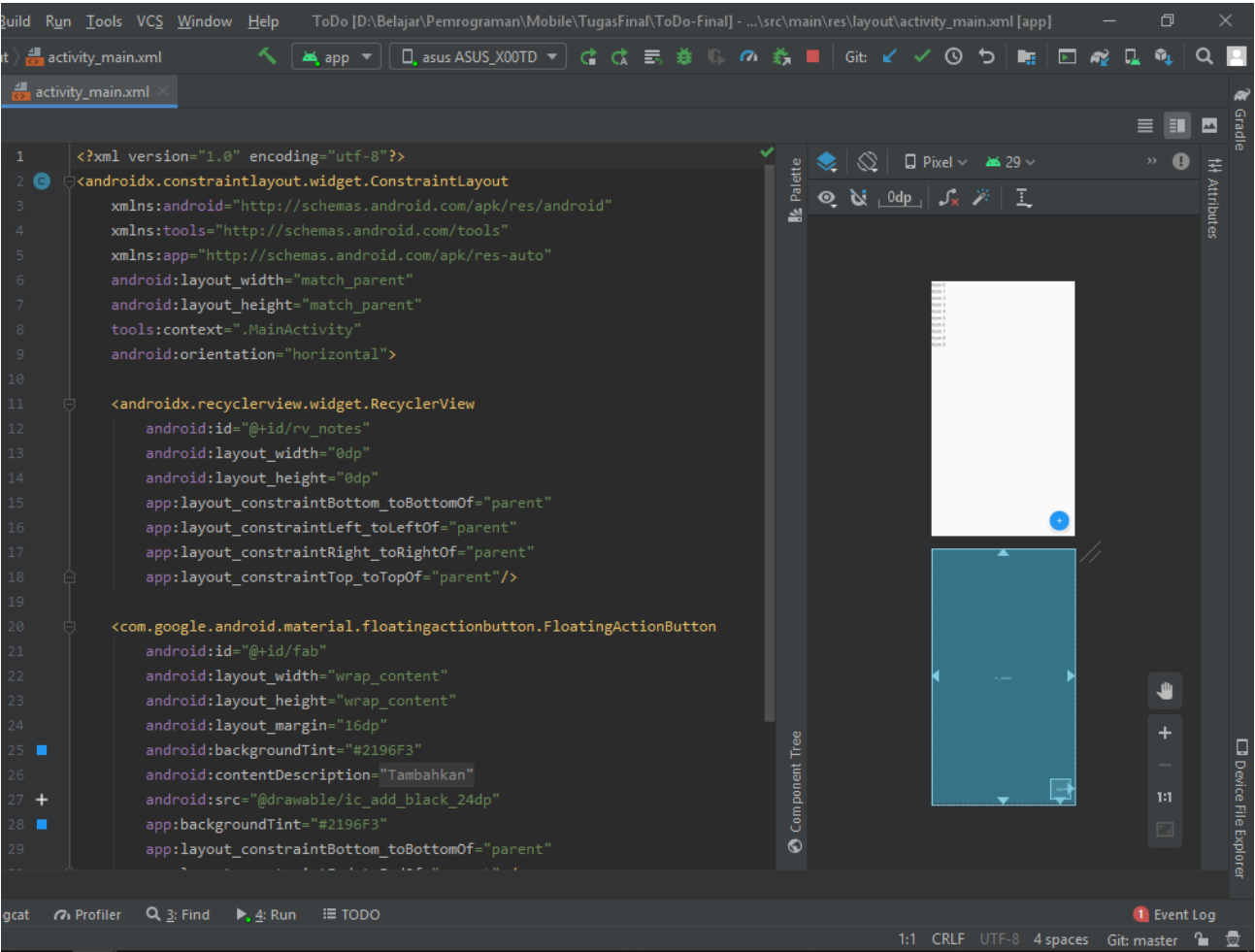
```
build.gradle (app) x build.gradle (ToDo) x
1  apply plugin: 'com.android.application'
2  apply plugin: 'kotlin-android'
3  apply plugin: 'kotlin-android-extensions'
4  apply plugin: 'kotlin-kapt'
5  apply plugin: 'kotlin-android-extensions'
6
7  android {
8      compileSdkVersion 29
9      buildToolsVersion "29.0.3"
10
11     defaultConfig {
12         applicationId "id.ac.unhas.todolist"
13         minSdkVersion 26
14         targetSdkVersion 29
15         versionCode 1
16         versionName "1.0"
17
18         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
19     }
20
21     buildTypes {
22         release {
23             minifyEnabled false
24             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
25         }
26     }
27
28 }
29
30 dependencies {
31     implementation fileTree(dir: 'libs', include: ['*.jar'])
    dependencies{
    }
}
```

build.gradle (:app) × build.gradle (ToDo) ×

```
29
30 dependencies {
31     implementation fileTree(dir: 'libs', include: ['*.jar'])
32     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
33     implementation 'androidx.appcompat:appcompat:1.1.0'
34     implementation 'androidx.core:core-ktx:1.2.0'
35     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
36     testImplementation 'junit:junit:4.13'
37     androidTestImplementation 'androidx.test.ext:junit:1.1.1'
38     androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
39
40     // Material design
41     implementation "com.google.android.material:material:1.1.0"
42
43     // RecyclerView
44     implementation "androidx.recyclerview:recyclerview:1.1.0"
45
46     // Coroutines
47     implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.4"
48
49     // Room
50     implementation "androidx.room:room-runtime:2.2.5"
51     implementation "androidx.room:room-ktx:2.2.5"
52     kapt "androidx.room:room-compiler:2.2.5"
53
54     // ViewModel
55     implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"
56     kapt "androidx.lifecycle:lifecycle-compiler:2.2.0"
57 }
```

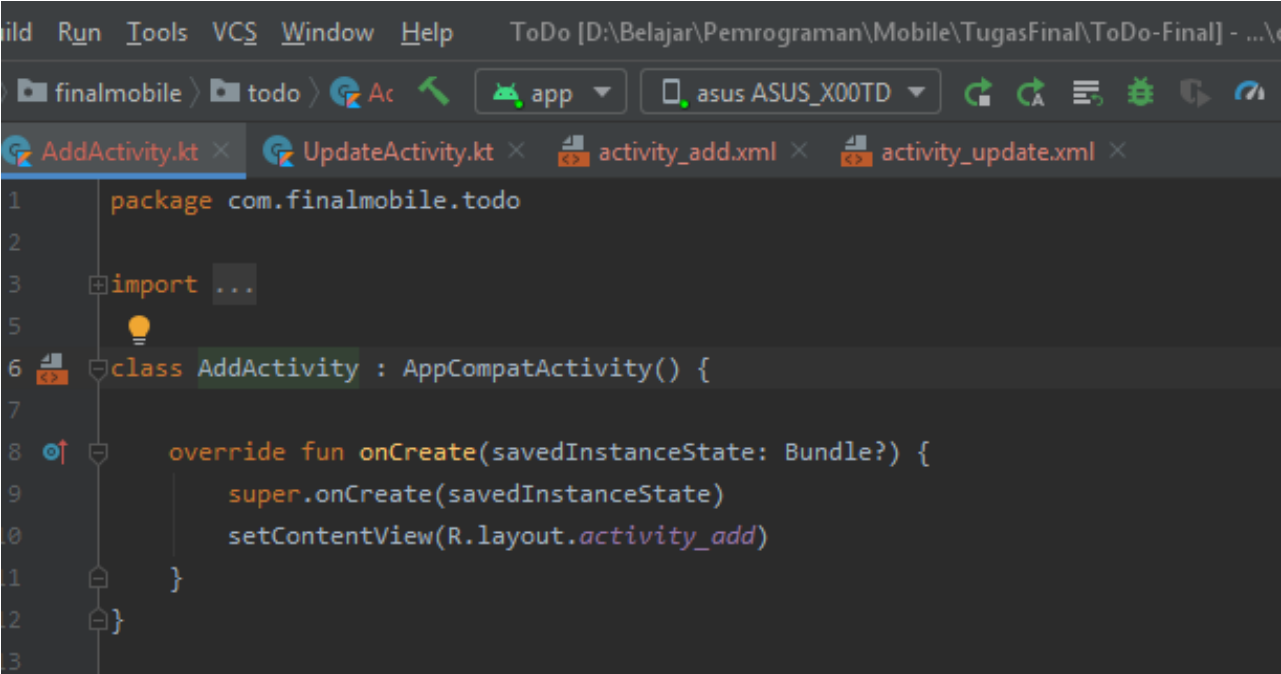

Mengubah Tampilan Di Activity Main

Pada bagian ini kita membuat desain UI dari activit_main kita. Didalamnya terdapat dua item yaitu recycler view untuk menampilkan daftar tugas aplikasi kita dan float button untuk membuat tugas baru. Pada bagian ini saya juga mengubah warna dasar dari aplikasi saya dengan tema berwarna biru

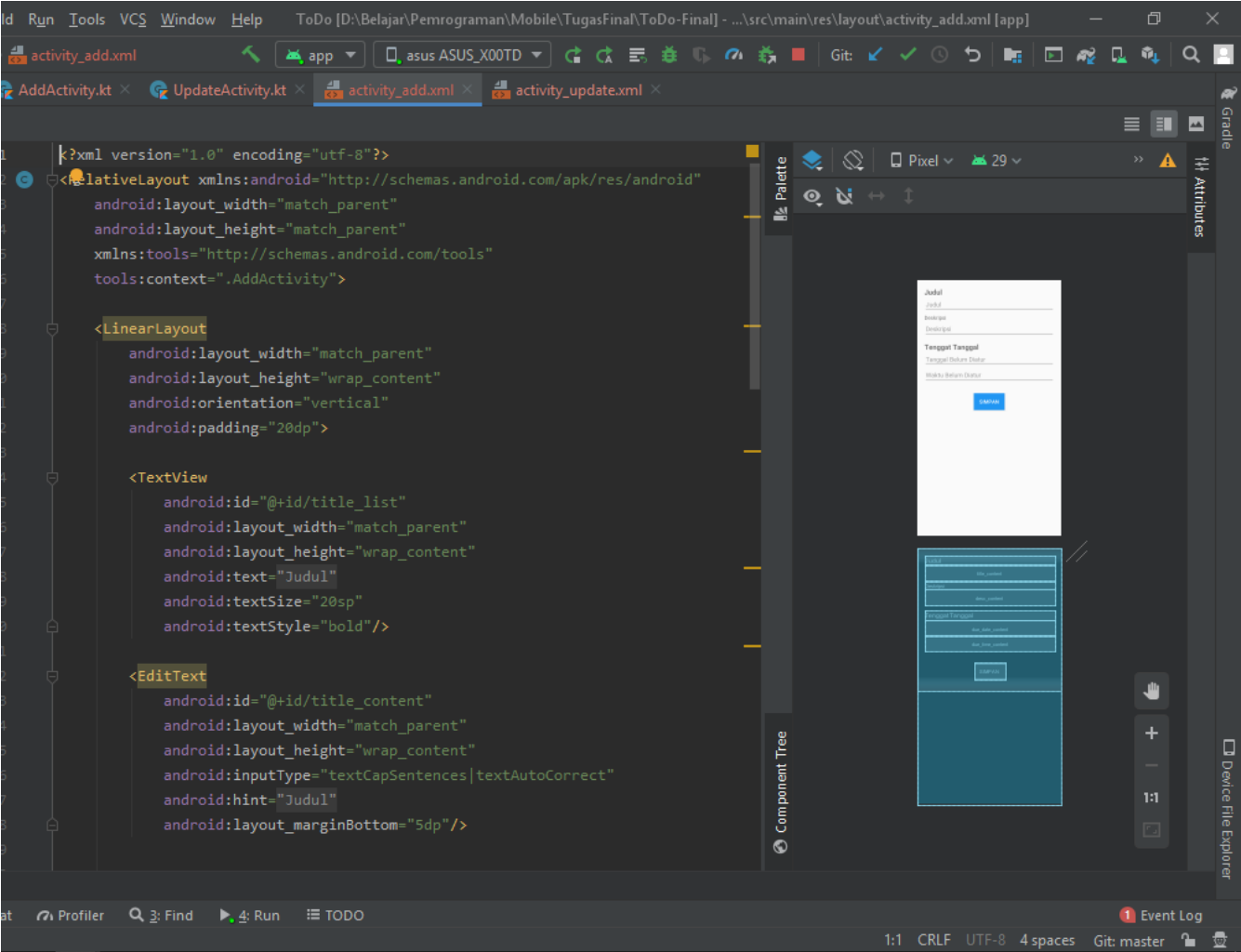


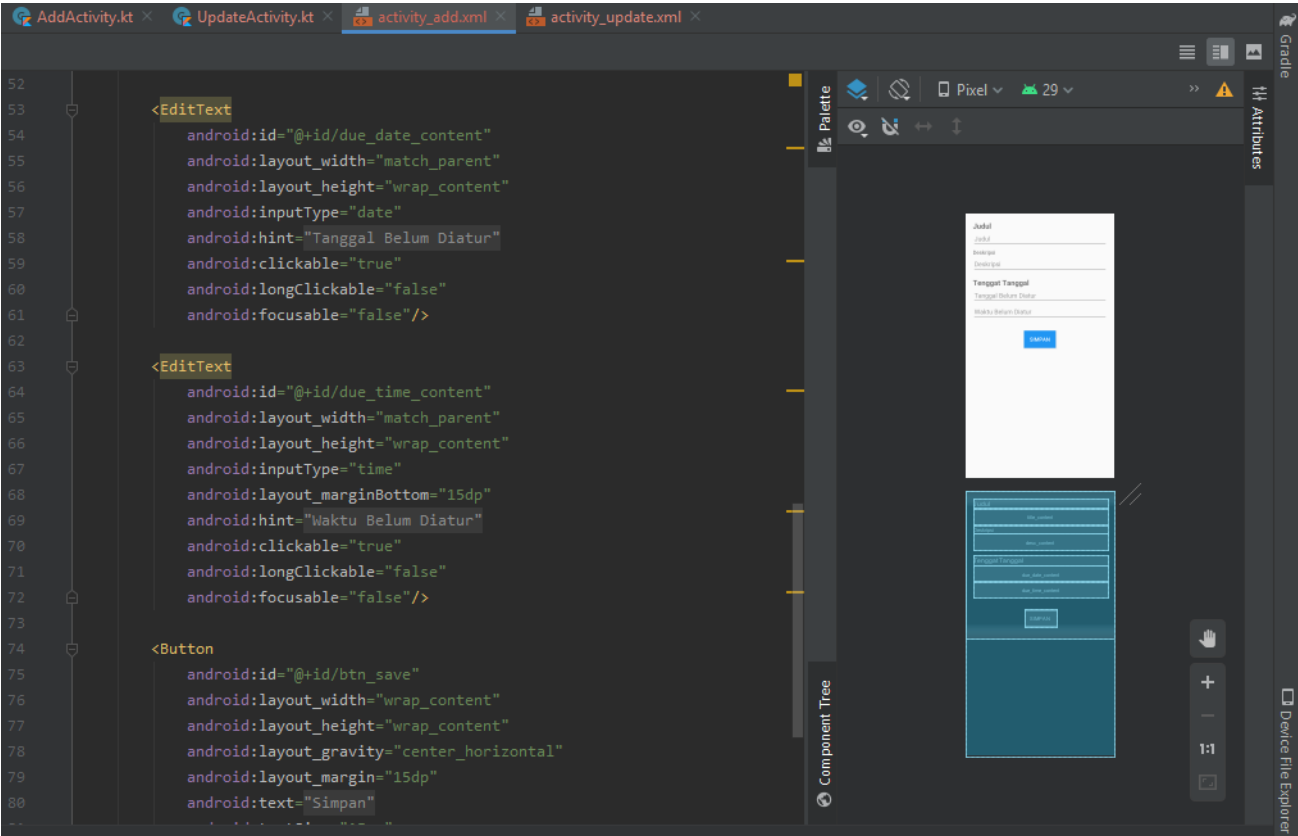
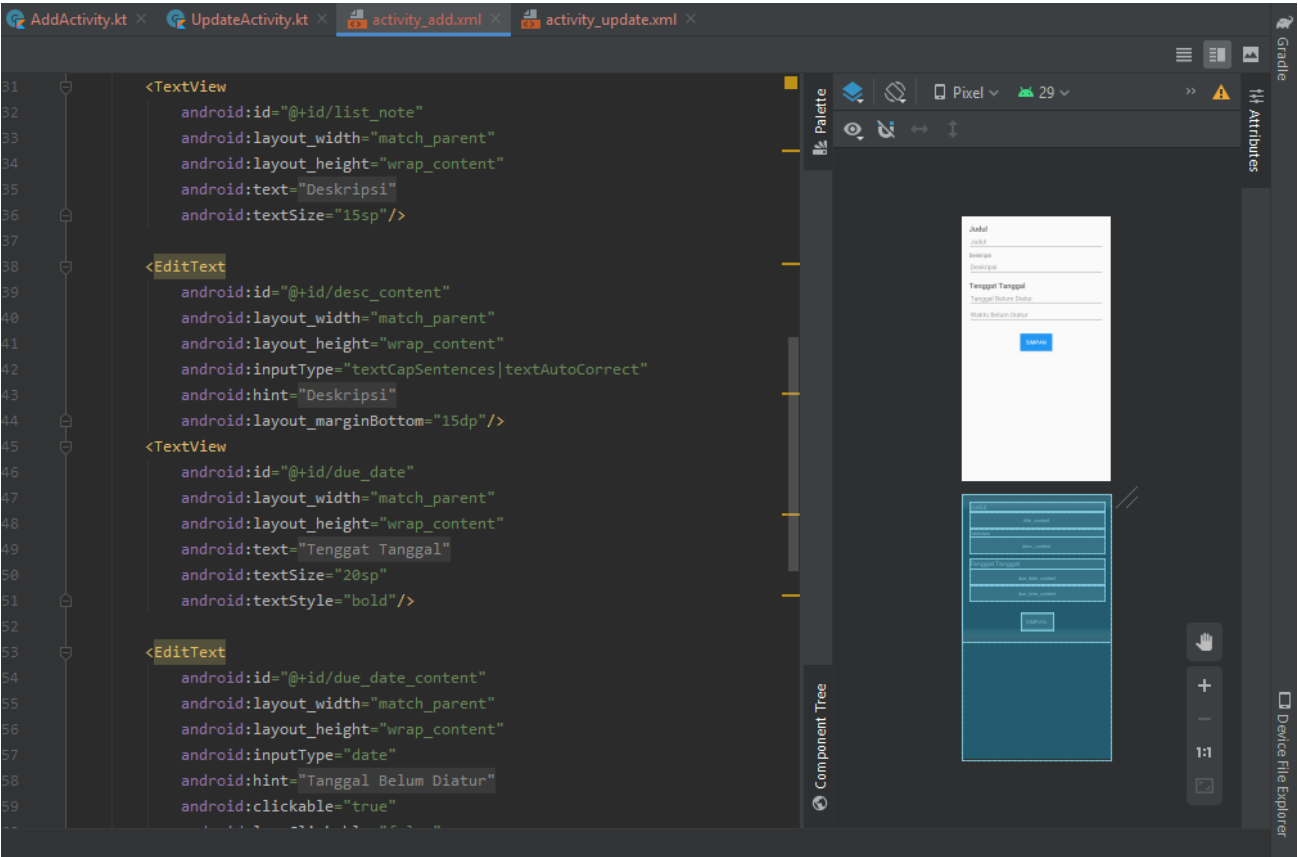
Menambahkan AddActivity

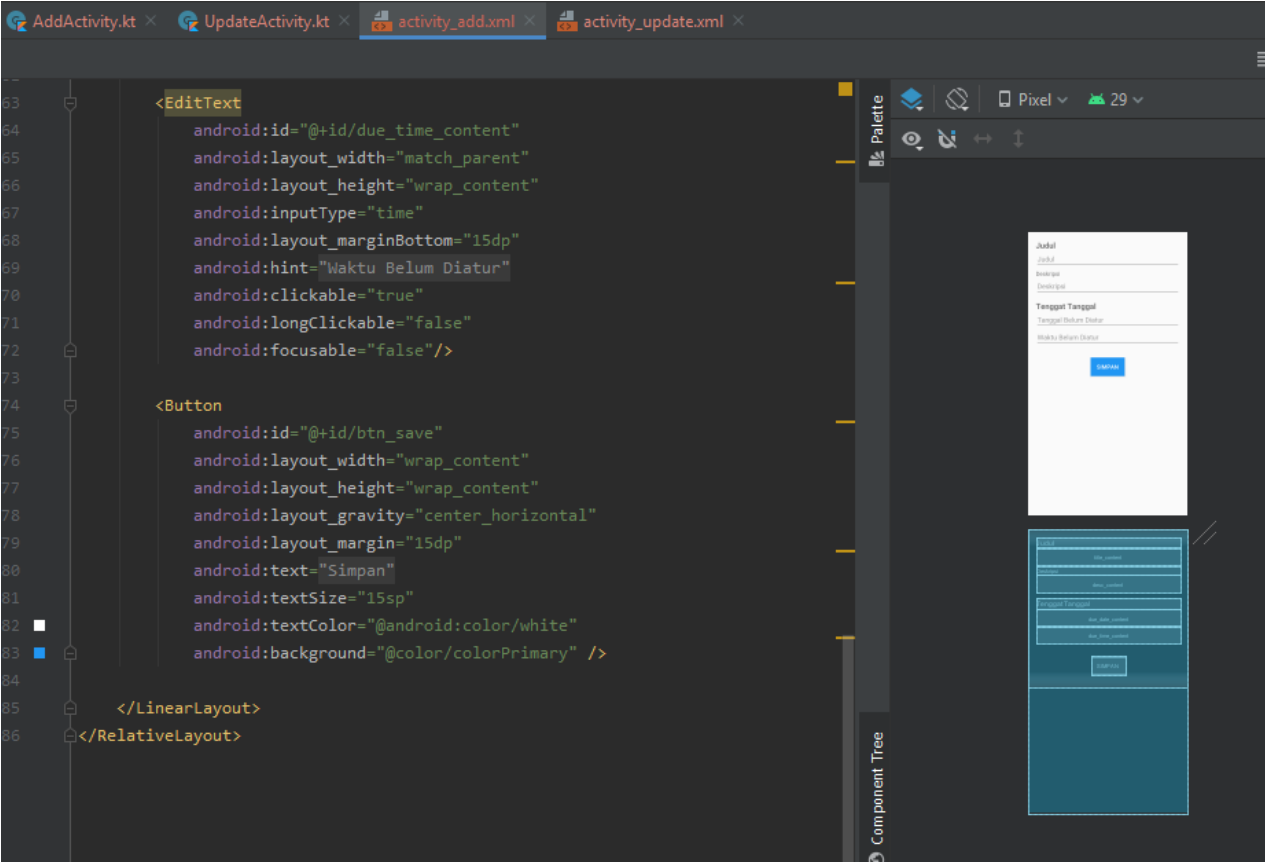
Kemudian Kita membuat activity baru yang bernama AddActivity untuk mengatur bagian aplikasi pada saat menambahkan tugas baru



Dibawah juga sudah saya atur tampilannya sesuai selera dan keperluan saya

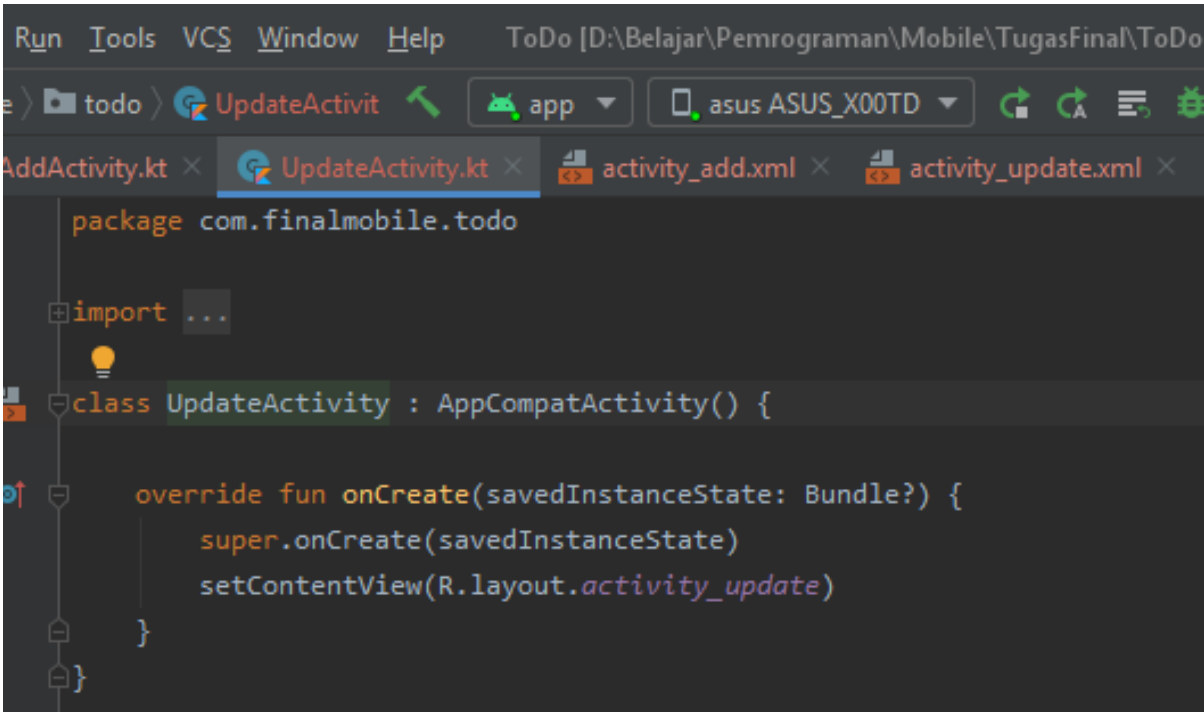




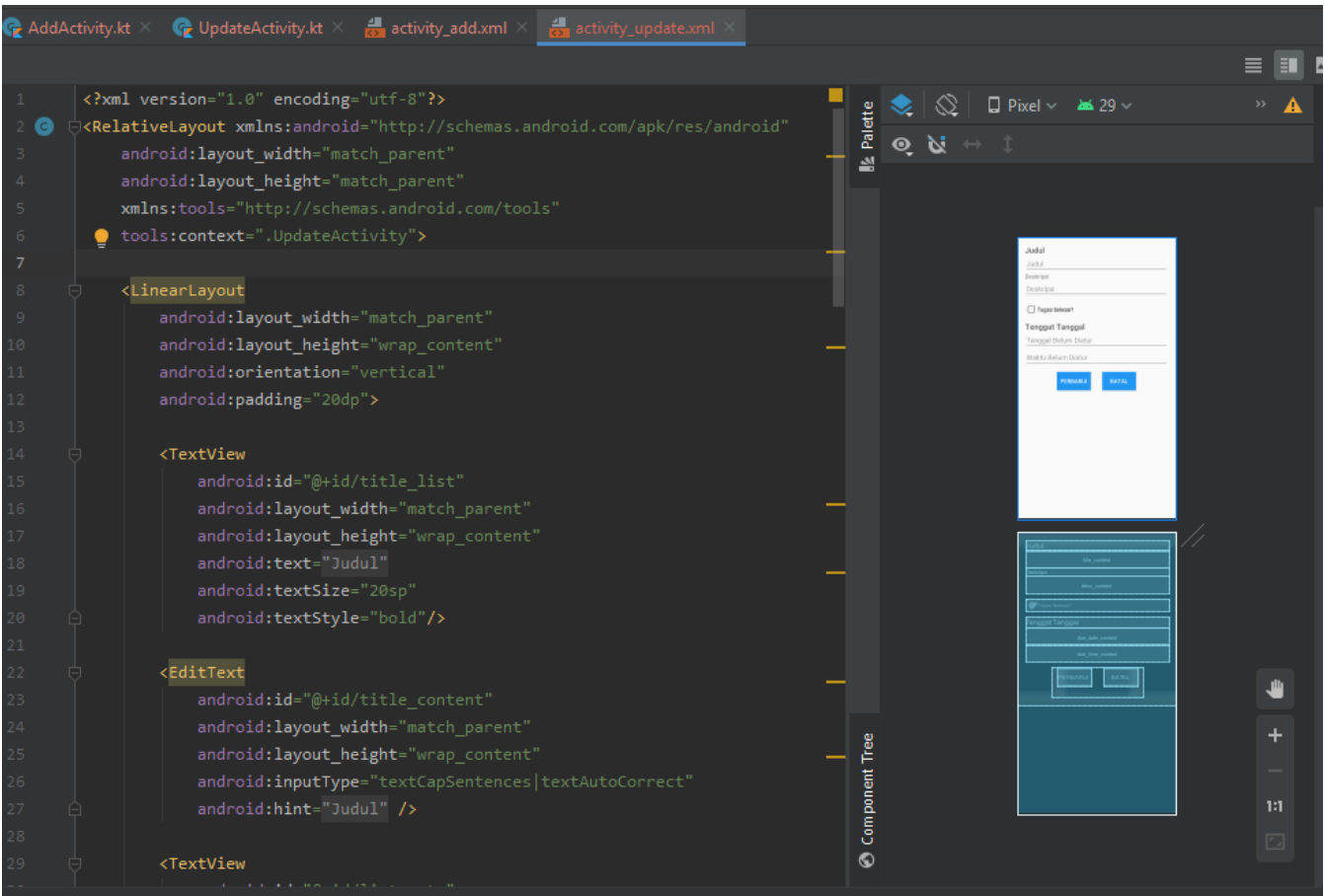


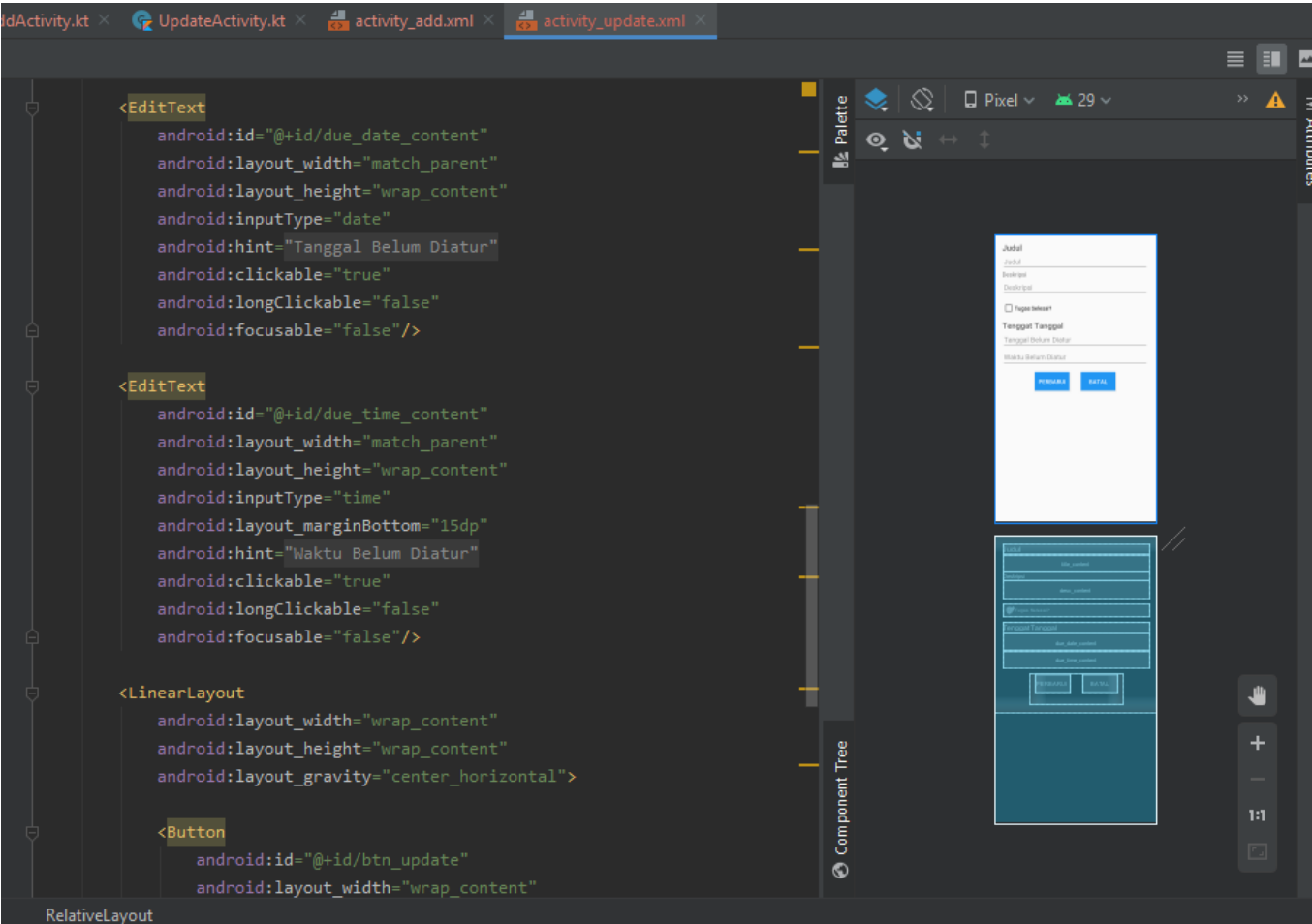
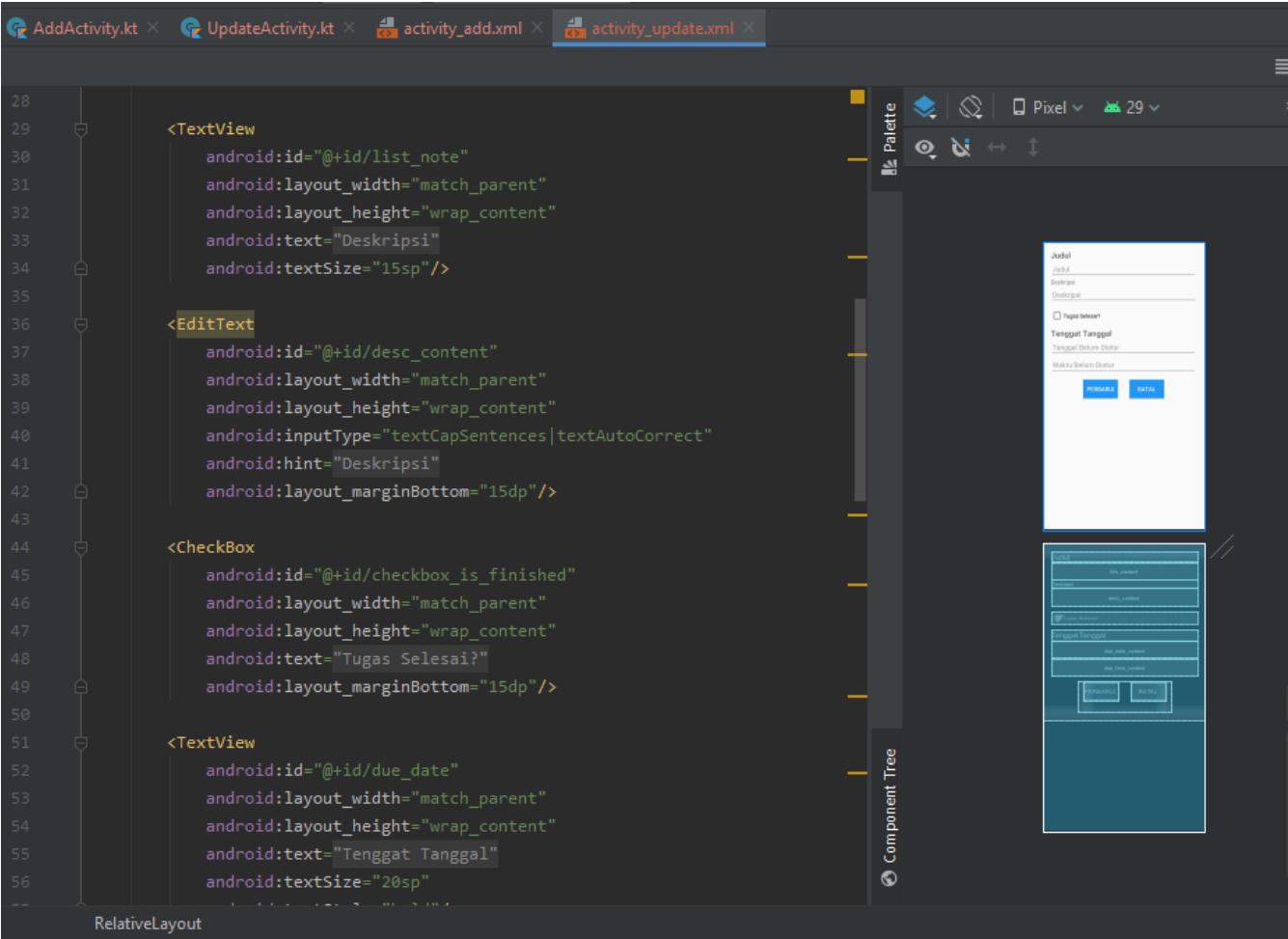
Menambahkan Update Activity

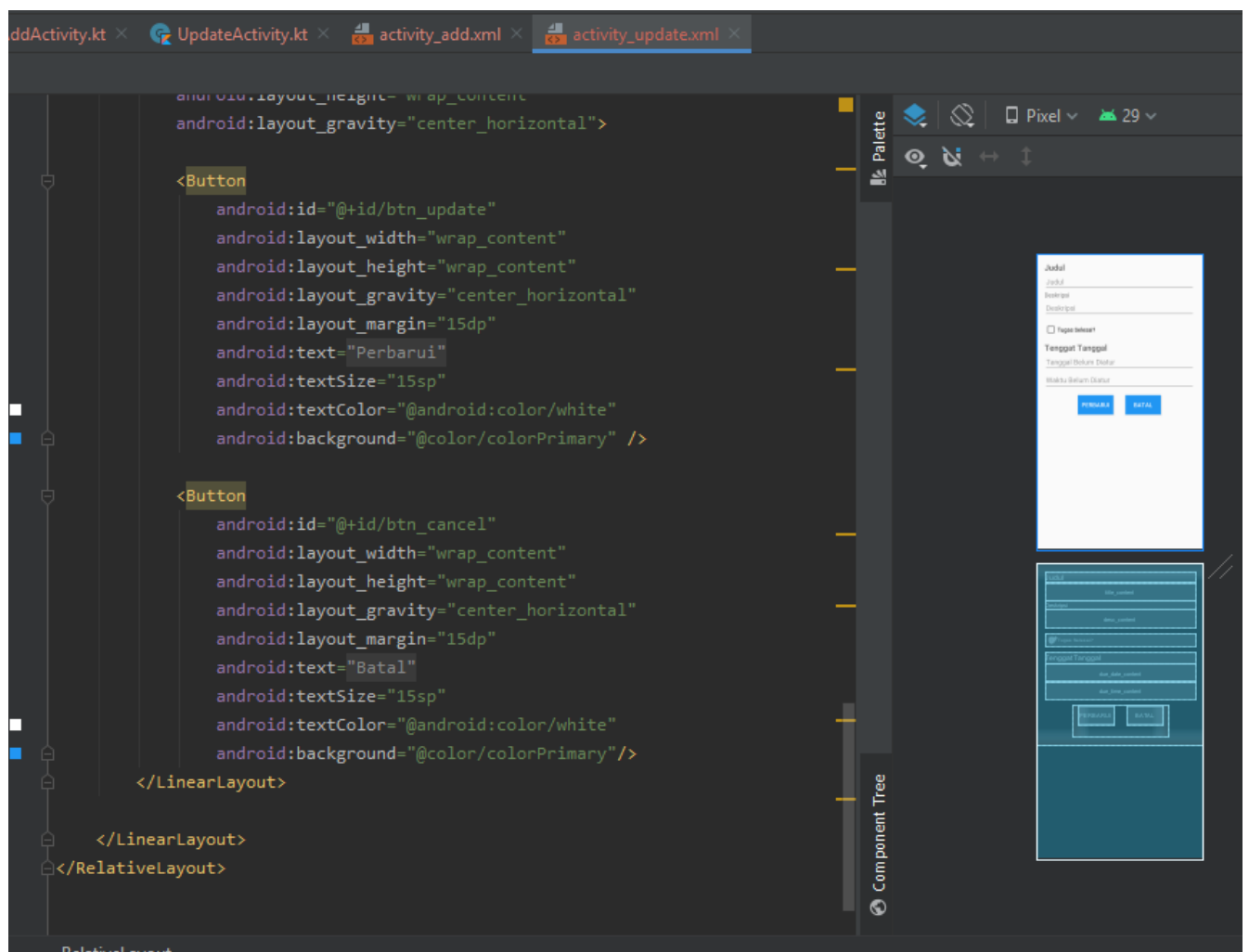
Kemudian saya membuat activity baru lagi yang bernama UpdateActivity untuk mengatur aplikasi pada saat ingin mengubah suatu tugas, menghapus, ataupun menyelesaikannya



Saya juga mengatur tampilannya sesuai selera dan keperluan saya







Menambahkan Entity Untuk Room

Kemudian, untuk membuat Database menggunakan Room, hal pertama yang kita lakukan adalah membuat sebuah Entity, dalam hal ini bernama ToDo untuk menyimpan values dari data kita. Didalam hal ini saya memiliki 11 kolom

```
1 package com.finalmobile.todo.database
2
3 import ...
4
5
6
7
8
9 @Parcelize
10 @Entity(tableName = "todolist")
11 data class ToDo (
12     @PrimaryKey(autoGenerate = true)
13     @ColumnInfo(name = "id")
14     val id: Int? = null,
15
16     @ColumnInfo(name = "created_date")
17     var createdDate: Int? = null,
18
19     @ColumnInfo(name = "str_created_date")
20     var strCreatedDate: String? = null,
21
22     @ColumnInfo(name = "updated_date")
23     var updatedAt: Int? = null,
24
25     @ColumnInfo(name = "due_date")
26     var dueDate: Int? = null,
27
28     @ColumnInfo(name = "due_hour")
29     var dueHour: Int? = null,
30
31     @ColumnInfo(name = "str_due_date")
32     var strDueDate: String? = null,
33
34     @ColumnInfo(name = "str_due_hour")
35     var strDueHour: String? = null,
```

```
20     var strCreatedDate: String? = null,
21
22     @ColumnInfo(name = "updated_date")
23     var updatedAt: Int? = null,
24
25     @ColumnInfo(name = "due_date")
26     var dueDate: Int? = null,
27
28     @ColumnInfo(name = "due_hour")
29     var dueHour: Int? = null,
30
31     @ColumnInfo(name = "str_due_date")
32     var strDueDate: String? = null,
33
34     @ColumnInfo(name = "str_due_hour")
35     var strDueHour: String? = null,
36
37     @ColumnInfo(name = "title")
38     var title: String,
39
40     @ColumnInfo(name = "note")
41     var note: String,
42
43     @ColumnInfo(name = "is_finished")
44     var isFinished: Boolean? = null
45 ) : Parcelable
```


Menambahkan Dao untuk Room

Kemudian, yang kedua adalah membuat Dao, untuk mengambil entity dari dalam database dalam hal ini saya memiliki 8 query.

```
package com.finalmobile.todo.database

import ...

@Dao
interface ToDoDao {

    @Query( value: "SELECT * FROM todolist ORDER BY due_date ASC, due_hour ASC")
    fun getToDoList(): LiveData<List<ToDo>>

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insertList(list: ToDo)

    @Delete
    suspend fun deleteList(list: ToDo)

    @Update
    suspend fun updateList(list: ToDo)

    @Query( value: "SELECT * FROM todolist WHERE title LIKE :title")
    fun searchResult(title: String): LiveData<List<ToDo>>

    @Query( value: "SELECT * FROM todolist ORDER BY due_date DESC, due_hour DESC")
    fun sortByDueDateDescending(): LiveData<List<ToDo>>

    @Query( value: "SELECT * FROM todolist ORDER BY created_date ASC")
    fun sortByCreatedDateAscending(): LiveData<List<ToDo>>

    @Query( value: "SELECT * FROM todolist ORDER BY created_date DESC")
    fun sortByCreatedDateDescending(): LiveData<List<ToDo>>
}
```

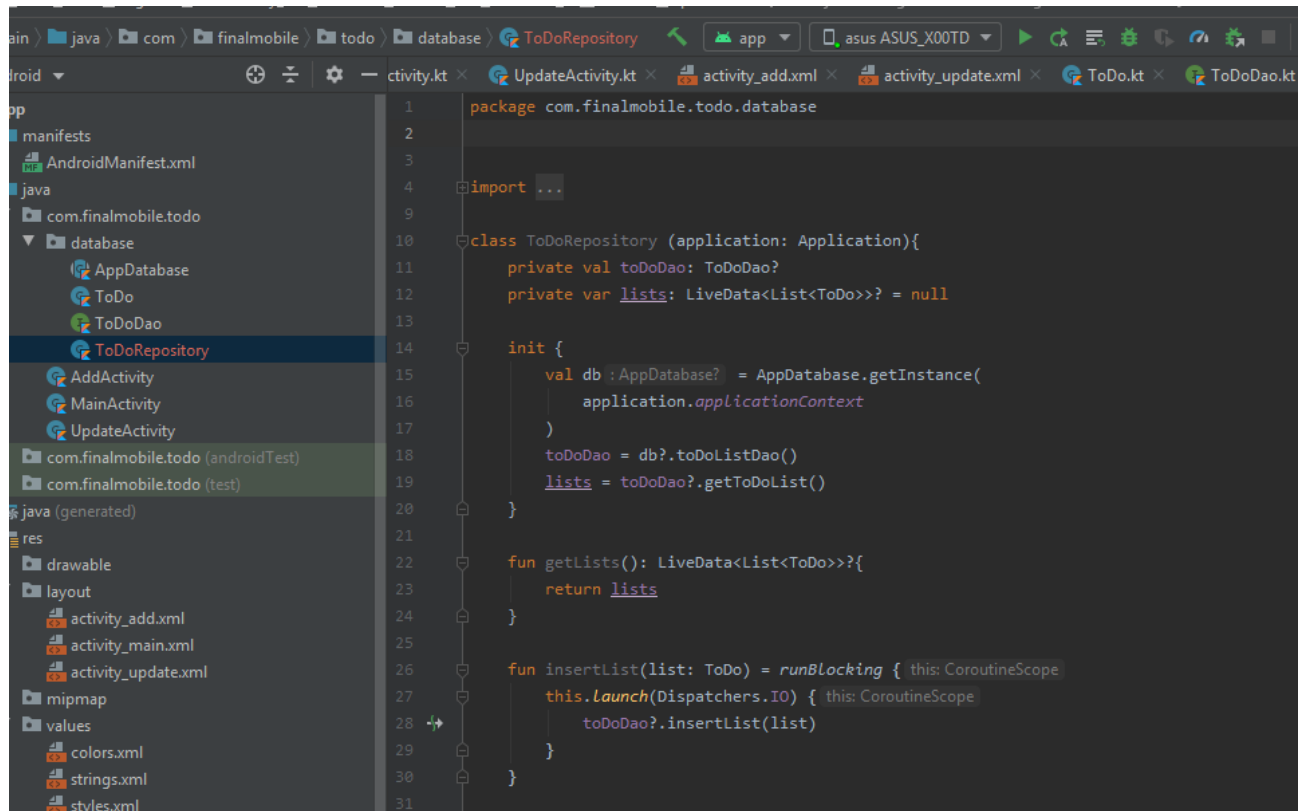
Membuat Database untuk Room

Selanjutnya kita membuat Database, untuk menyimpan data data yang di input kedalam aplikasi

```
1 package com.finalmobile.todo.database
2
3 import android.content.Context
4 import androidx.room.Database
5 import androidx.room.Room
6 import androidx.room.RoomDatabase
7
8 @Database(entities = [ToDo::class], exportSchema = false, version = 1)
9 abstract class AppDatabase : RoomDatabase(){
10     abstract fun toDoListDao(): ToDoDao
11
12     companion object{
13         private const val DB_NAME = "TO_DO_LIST_DB"
14         private var instance: AppDatabase? = null
15
16         fun getInstance(context: Context): AppDatabase?{
17             if(instance == null){
18                 synchronized(AppDatabase::class){
19                     instance = Room
20                         .databaseBuilder(
21                             context,
22                             AppDatabase::class.java,
23                             DB_NAME
24                         )
25                         .build()
26                 }
27             }
28             return instance
29         }
30     }
31 }
```

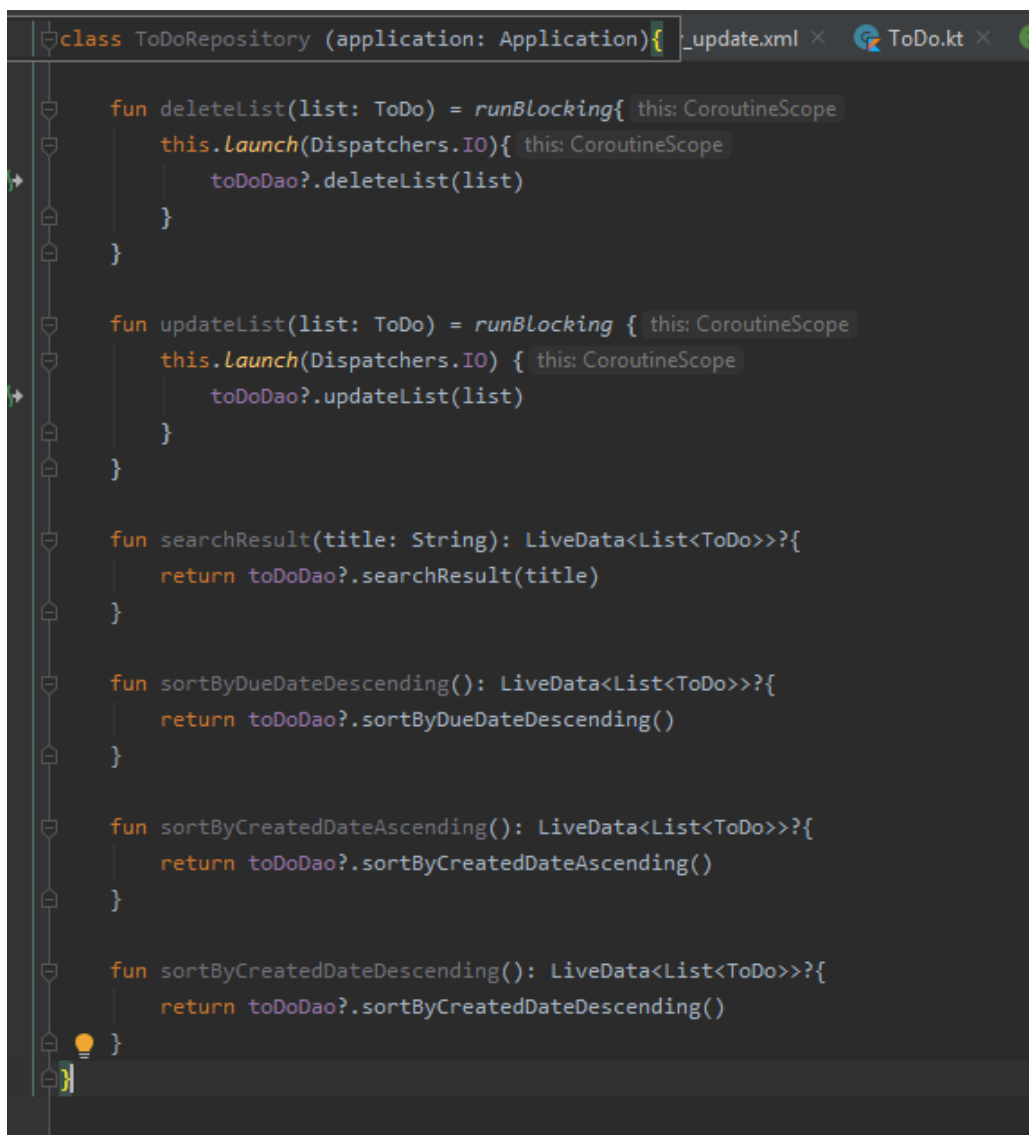
Membuat Repository

Kemudian kita membuat sebuah repository untuk menghubungkan view model dengan database Dao kita. Ke empat file tadi kemudian saya masukkan kedalam satu package dengan nama database



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes a package named `com.finalmobile.todo.database` containing `AppDatabase`, `ToDo`, `ToDoDao`, and `ToDoRepository`. The code editor shows the `ToDoRepository` class with the following code:

```
1 package com.finalmobile.todo.database
2
3
4 import ...
5
6
7
8
9
10 class ToDoRepository (application: Application){
11     private val toDoDao: ToDoDao?
12     private var lists: LiveData<List<ToDo>>? = null
13
14     init {
15         val db :AppDatabase? = AppDatabase.getInstance(
16             application.applicationContext
17         )
18         toDoDao = db?.toDoListDao()
19         lists = toDoDao?.getToDoList()
20     }
21
22     fun getLists(): LiveData<List<ToDo>>?{
23         return lists
24     }
25
26     fun insertList(list: ToDo) = runBlocking { this: CoroutineScope
27         this.launch(Dispatchers.IO) { this: CoroutineScope
28             toDoDao?.insertList(list)
29         }
30     }
31 }
```



The screenshot shows the continuation of the `ToDoRepository` class code in the IDE. The code includes methods for deleting, updating, searching, and sorting the list of todos.

```
class ToDoRepository (application: Application){
    fun deleteList(list: ToDo) = runBlocking{ this: CoroutineScope
        this.launch(Dispatchers.IO){ this: CoroutineScope
            toDoDao?.deleteList(list)
        }
    }

    fun updateList(list: ToDo) = runBlocking { this: CoroutineScope
        this.launch(Dispatchers.IO) { this: CoroutineScope
            toDoDao?.updateList(list)
        }
    }

    fun searchResult(title: String): LiveData<List<ToDo>>?{
        return toDoDao?.searchResult(title)
    }

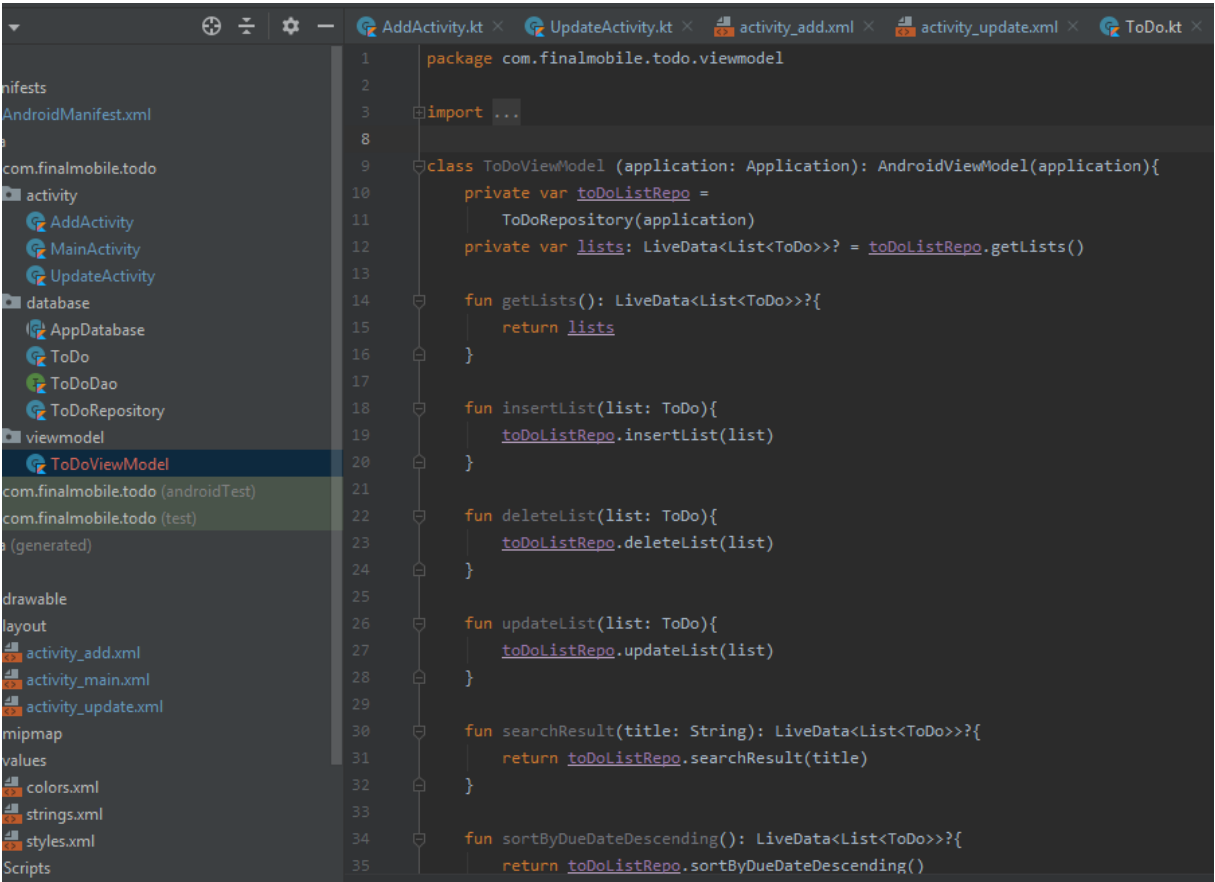
    fun sortByDueDateDescending(): LiveData<List<ToDo>>?{
        return toDoDao?.sortByDueDateDescending()
    }

    fun sortByCreatedDateAscending(): LiveData<List<ToDo>>?{
        return toDoDao?.sortByCreatedDateAscending()
    }

    fun sortByCreatedDateDescending(): LiveData<List<ToDo>>?{
        return toDoDao?.sortByCreatedDateDescending()
    }
}
```

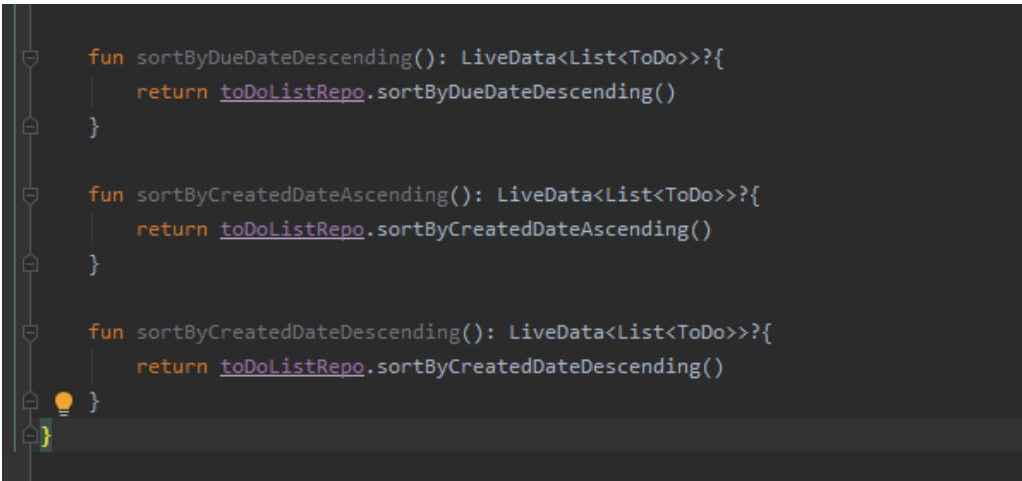
Membuat View Model

Kemudian untuk membuat method method dari aplikasi antara database dengan activity dari aplikasi, kita membuat view model. View model ini kemudian saya masukkan kedalam package viewmodel



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes packages like `com.finalmobile.todo` with sub-packages `activity`, `database`, and `viewmodel`. The `viewmodel` package contains `ToDoViewModel`. The code editor shows the implementation of `ToDoViewModel` in `ToDo.kt`.

```
1 package com.finalmobile.todo.viewmodel
2
3 import ...
4
5
6
7
8
9 class ToDoViewModel (application: Application): AndroidViewModel(application){
10     private var toDoListRepo =
11         ToDoRepository(application)
12     private var lists: LiveData<List<ToDo>>? = toDoListRepo.getLists()
13
14     fun getLists(): LiveData<List<ToDo>>?{
15         return lists
16     }
17
18     fun insertList(list: ToDo){
19         toDoListRepo.insertList(list)
20     }
21
22     fun deleteList(list: ToDo){
23         toDoListRepo.deleteList(list)
24     }
25
26     fun updatelist(list: ToDo){
27         toDoListRepo.updateList(list)
28     }
29
30     fun searchResult(title: String): LiveData<List<ToDo>>?{
31         return toDoListRepo.searchResult(title)
32     }
33
34     fun sortByDueDateDescending(): LiveData<List<ToDo>>?{
35         return toDoListRepo.sortByDueDateDescending()
```



This is a close-up view of the code editor showing the implementation of three sorting methods in the `ToDoViewModel` class. Each method returns a `LiveData<List<ToDo>>?` object by delegating the call to the `toDoListRepo`.

```
fun sortByDueDateDescending(): LiveData<List<ToDo>>?{
    return toDoListRepo.sortByDueDateDescending()
}

fun sortByCreatedDateAscending(): LiveData<List<ToDo>>?{
    return toDoListRepo.sortByCreatedDateAscending()
}

fun sortByCreatedDateDescending(): LiveData<List<ToDo>>?{
    return toDoListRepo.sortByCreatedDateDescending()
}
```

Menghubungkan MainActivity dengan AddActivity

Selanjutnya kita membuat MainActivity dan AddActivity kita terhubung menggunakan intent dengan perantara FloatingActionButton. Saya juga membuat semua activity menjadi satu folder

▼ activity

AddActivity

MainActivity

UpdateActivity

▼ database

AppDatabase

ToDo

ToDoDao

ToDoRepository

▼ viewmodel

ToDoViewModel

com.finalmobile.todo (androidTest)

com.finalmobile.todo (test)

java (generated)

res

drawable

layout

activity_add.xml

activity_main.xml

activity_update.xml

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.finalmobile.todo.R
import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity() {

    private lateinit var floatingActionButton: FloatingActionButton

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        floatingActionButton = findViewById(R.id.fab)

        floatingActionButton.setOnClickListener { it: View!
            addList()
        }
    }
}
```

Membuat Fitur Alert Saat Menekan Salah Satu Tugas

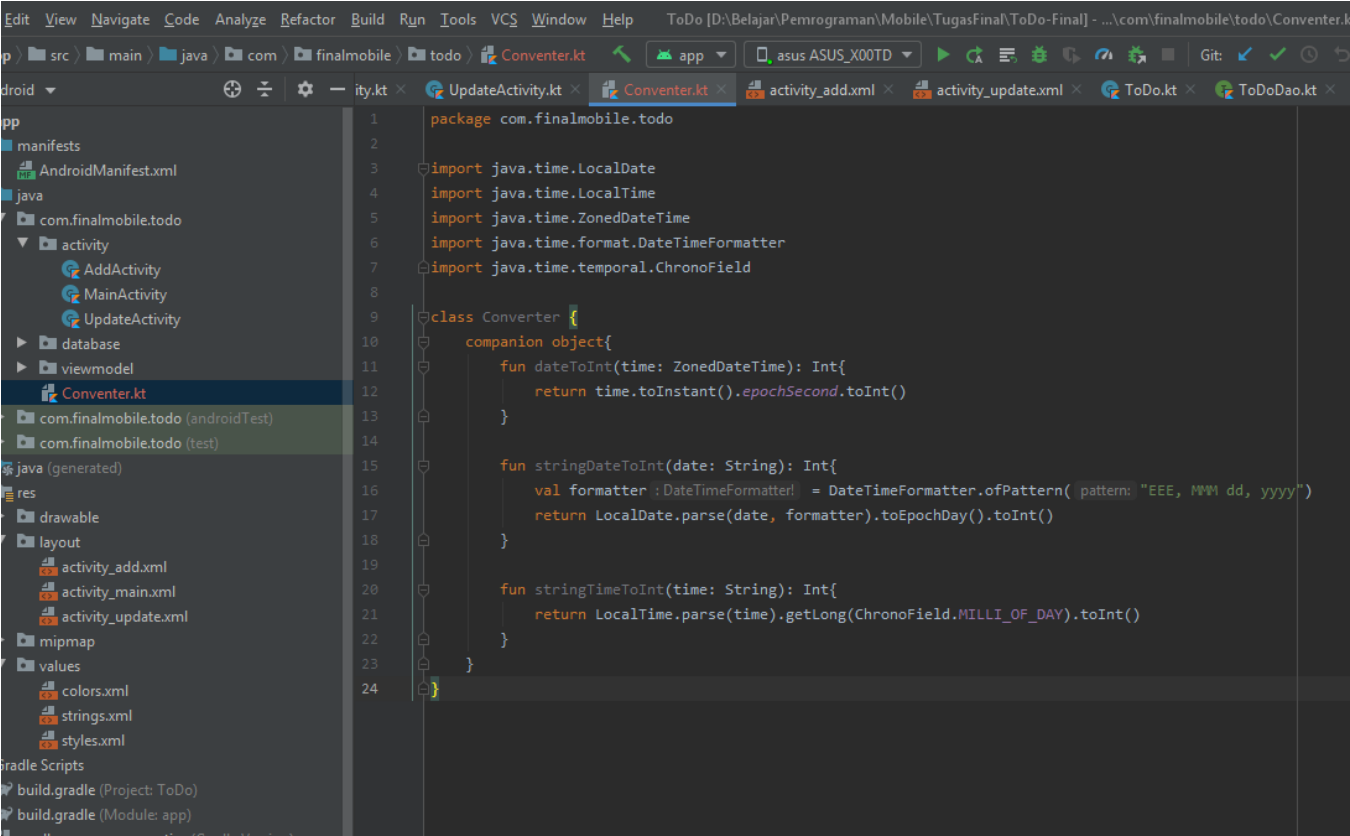
Fitur ini dibuat di dalam MainActivity, nanti kita gunakan untuk memunculkan menu pada saat menekan salah satu tugas, dengan menu yaitu detail tugas, perbarui tugas, dan hapus tugas

```
private fun showAlertMenu(todo: ToDo){
    val items : Array<String> = arrayOf("Rincian", "Ubah", "Hapus")

    val builder = AlertDialog.Builder( context: this)
    val alert = AlertDialog.Builder( context: this)
    builder.setItems(items){ dialog, which ->
        when(which){
            0 -> {
                //list
            }
            1 -> {
                //Update
            }
            2 -> {
                alert.setTitle("Hapus Tugas?")
                .setMessage("Tugas yang dihapus tidak dapat dikembalikan, kamu yakin?")
                .setPositiveButton( text: "Iya"){dialog, _ ->
                    toDoViewModel.deletelist(todo)
                    dialog.dismiss()
                }
                .setNegativeButton( text: "Tidak"){dialog, _ ->
                    dialog.dismiss()
                }
                .show()
            }
        }
    }
    builder.show()
}
```

Membuat File Converter untuk mengubah waktu dari String ke Int

Kemudian kita membuat sebuah file converter untuk mengubah waktu yang kita input pada saat membuat tugas baru. Karena file yang kita input bertipe String, maka kita harus mengkonversinya menjadi Integer untuk disimpan kedalam Database



```
1 package com.finalmobile.todo
2
3 import java.time.LocalDate
4 import java.time.LocalDateTime
5 import java.time.ZonedDateTime
6 import java.time.format.DateTimeFormatter
7 import java.time.temporal.ChronoField
8
9 class Converter {
10     companion object {
11         fun dateToInt(time: ZonedDateTime): Int {
12             return time.toInstant().epochSecond.toInt()
13         }
14
15         fun stringDateToInt(date: String): Int {
16             val formatter : DateTimeFormatter! = DateTimeFormatter.ofPattern( pattern: "EEE, MMM dd, yyyy")
17             return LocalDate.parse(date, formatter).toEpochDay().toInt()
18         }
19
20         fun stringTimeToInt(time: String): Int {
21             return LocalDateTime.parse(time).getLong(ChronoField.MILLI_OF_DAY).toInt()
22         }
23     }
24 }
```

Menambahkan Isi AddActivity dan Menyesuaikan dengan Layout

Kemudian kita melengkapi isi dari AddActivity kita

```
Build Run Tools VCS Window Help  ToDo [D:\Belajar\Pemrograman\Mobile\TugasFinal\ToDo-Final]
todo > activity > AddActivity  app  asus ASUS_X00TD  activity_add.xml  activity_u

AddActivity.kt  UpdateActivity.kt  Converter.kt  activity_add.xml  activity_u

1  package com.finalmobile.todo.activity
2
3  import android.app.DatePickerDialog
4  import android.app.TimePickerDialog
5  import android.os.Bundle
6  import android.view.MenuItem
7  import android.widget.Button
8  import android.widget.EditText
9  import androidx.appcompat.app.AppCompatActivity
10 import androidx.lifecycle.ViewModelProvider
11 import com.finalmobile.todo.Converter
12 import com.finalmobile.todo.R
13 import com.finalmobile.todo.database.ToDo
14 import com.finalmobile.todo.viewmodel.ToDoViewModel
15 import java.text.SimpleDateFormat
16 import java.time.ZoneId
17 import java.time.ZonedDateTime
18 import java.time.format.DateTimeFormatter
19 import java.util.*
20
21 class AddActivity : AppCompatActivity() {
22     private lateinit var editTextTitle: EditText
23     private lateinit var editTextDate: EditText
24     private lateinit var editTextNote: EditText
25     private lateinit var editTextTime: EditText
26     private lateinit var btnSave: Button
27     private lateinit var todoViewModel: ToDoViewModel
28     private var calendar: Calendar = Calendar.getInstance()
29
30     override fun onCreate(savedInstanceState: Bundle?) {
31         super.onCreate(savedInstanceState)
```

```
Run Tools VCS Window Help  ToDo [D:\Belajar\Pemrograman\Mobile\TugasFinal\ToDo-Final] - ...finalmobile\todo\va
activity > AddActivity  app  asus ASUS_X00TD  activity_add.xml  activity_u
AddActivity.kt  UpdateActivity.kt  Converter.kt  activity_add.xml  activity_update.xml  ToD

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_add)
    if(supportActionBar != null){
        supportActionBar?.title = getString(R.string.tambah_tugas)
    }
    supportActionBar?.setDisplayHomeAsUpEnabled(true) // Back Button

    editTextTitle = findViewById(R.id.title_content)
    editTextDate = findViewById(R.id.due_date_content)
    editTextNote = findViewById(R.id.desc_content)
    editTextTime = findViewById(R.id.due_time_content)
    btnSave = findViewById(R.id.btn_save)
    todoViewModel = ViewModelProvider( owner: this).get(ToDoViewModel::class.java)

    editTextDate.setOnClickListener { it: View!
        setDueDate()
    }
    editTextTime.setOnClickListener { it: View!
        setDueTime()
    }
    btnSave.setOnClickListener { it: View!
        saveList()
    }
}
// Back Button
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    finish()
    return true
}
```

Method onOptionsItemSelected berfungsi untuk menyelesaikan AddActivity dan mengembalikannya ke MainActivity saat menekan tombol Back


```
Tools VCS Window Help ToDo [D:\Belajar\Pemrograman\Mobile\TugasFinal\ToDo-Final] - ...finalmobile\todo\activity\AddActivi
activity) AddActivity app asus ASUS_X00TD Git:
activity.kt UpdateActivity.kt Converter.kt activity_add.xml activity_update.xml ToDo.kt To
}

private fun setDueDate(){
    val date :Int = calendar.get(Calendar.DAY_OF_MONTH)
    val month :Int = calendar.get(Calendar.MONTH)
    val year :Int = calendar.get(Calendar.YEAR)

    // Date picker dialog
    val dateListener = DatePickerDialog.OnDateSetListener{ view, year, month, date ->
        calendar.set(Calendar.YEAR, year)
        calendar.set(Calendar.MONTH, month)
        calendar.set(Calendar.DATE, date)
        editTextDate.setText(SimpleDateFormat( pattern: "EEE, MMM dd, yyyy").format(calendar.time))
    }

    DatePickerDialog( context: this, dateListener, year, month, date).show()
}

private fun setDueTime(){
    val hour :Int = calendar.get(Calendar.HOUR_OF_DAY)
    val minute :Int = calendar.get(Calendar.MINUTE)

    val timeSetListener = TimePickerDialog.OnTimeSetListener { timePicker, hour, minute ->
        calendar.set(Calendar.HOUR_OF_DAY, hour)
        calendar.set(Calendar.MINUTE, minute)
        editTextTime.setText(SimpleDateFormat( pattern: "HH:mm").format(calendar.time))
    }

    TimePickerDialog( context: this, timeSetListener, hour, minute, is24HourView: true).show()
}

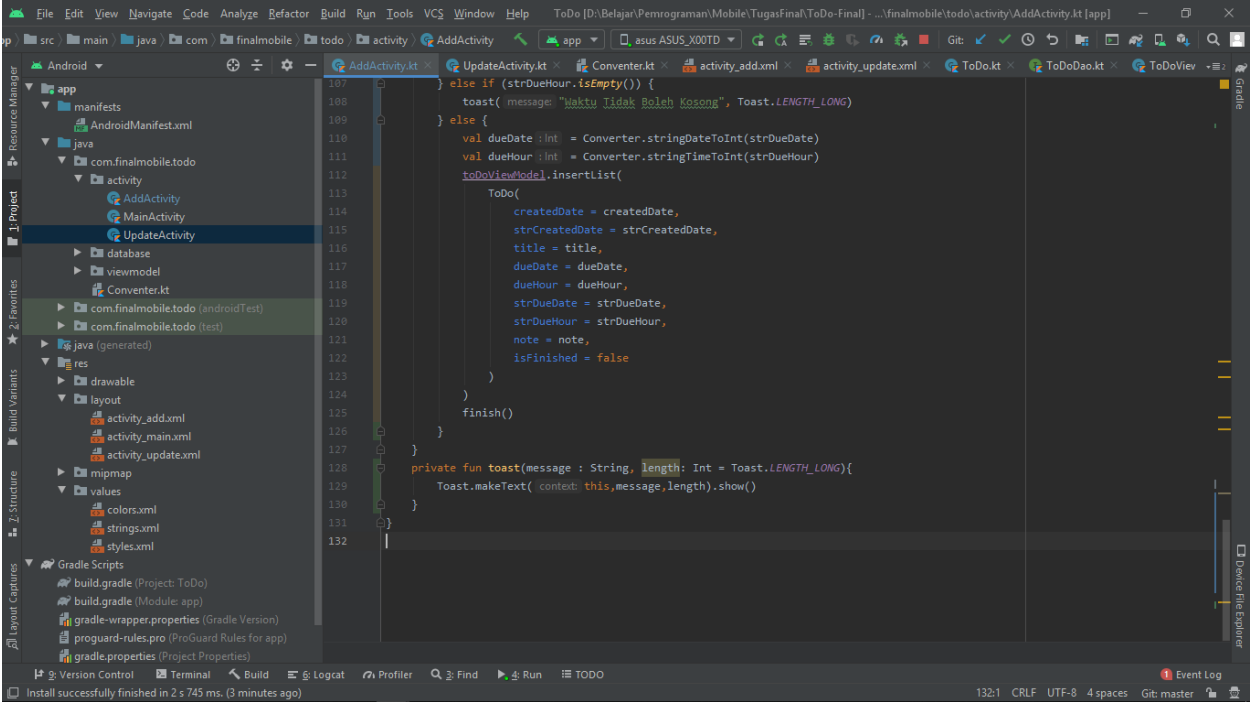
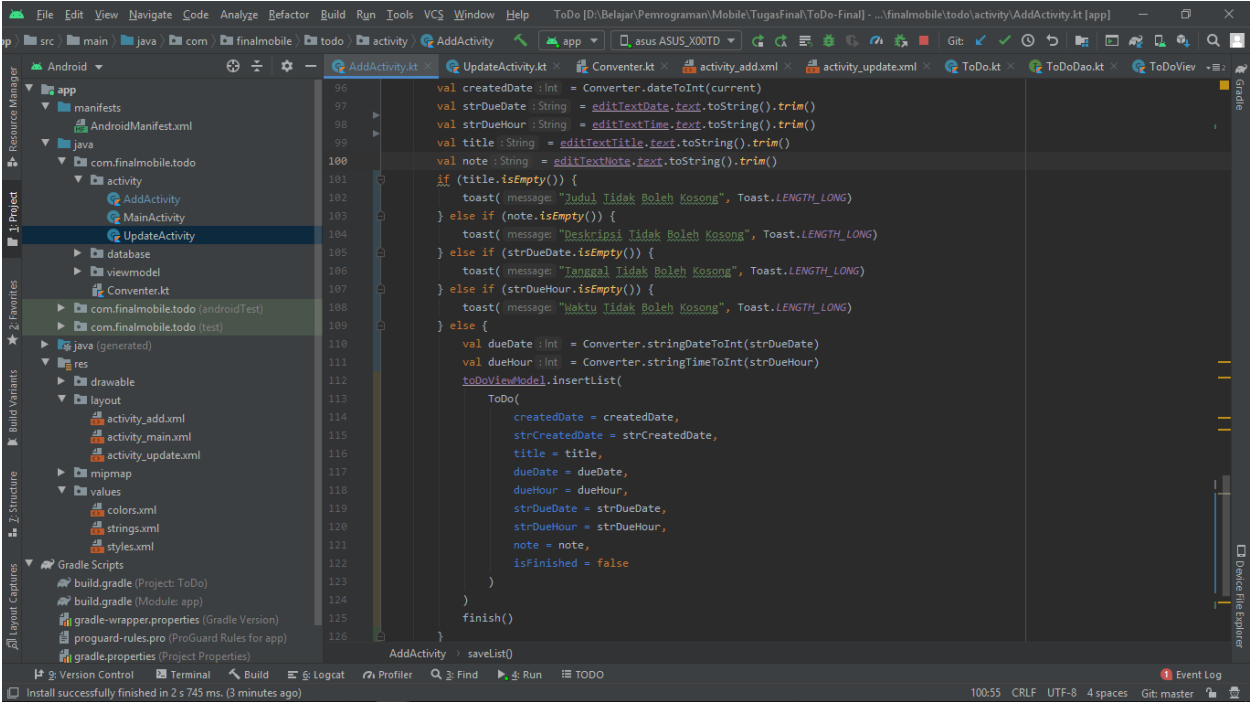
AddActivity
Profiler 3: Find 4: Run TODO 60:1 CRLF UT
```

```
private fun saveList(){

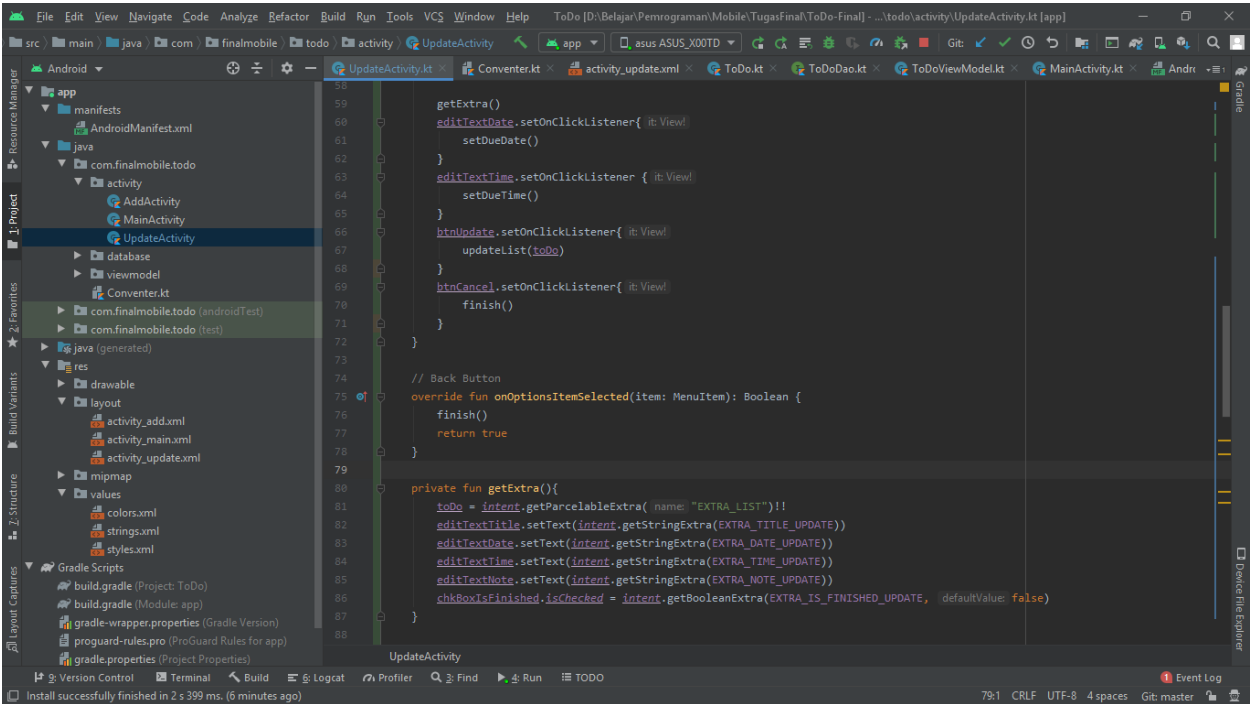
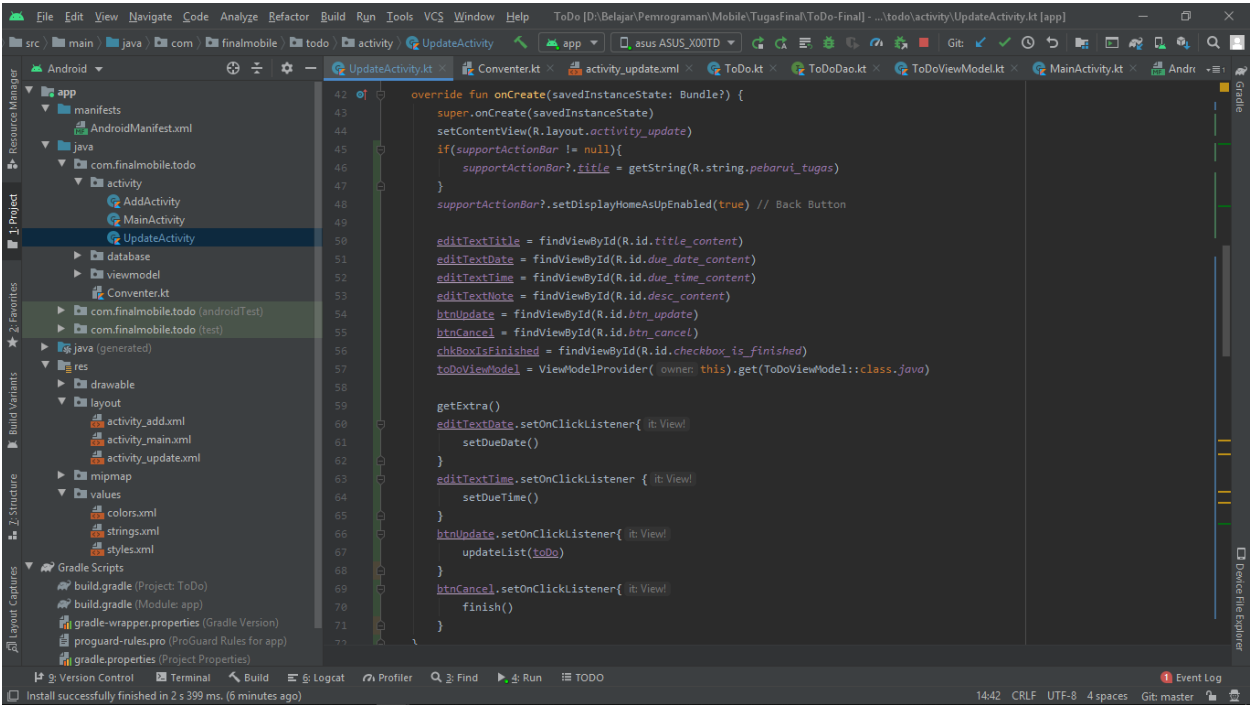
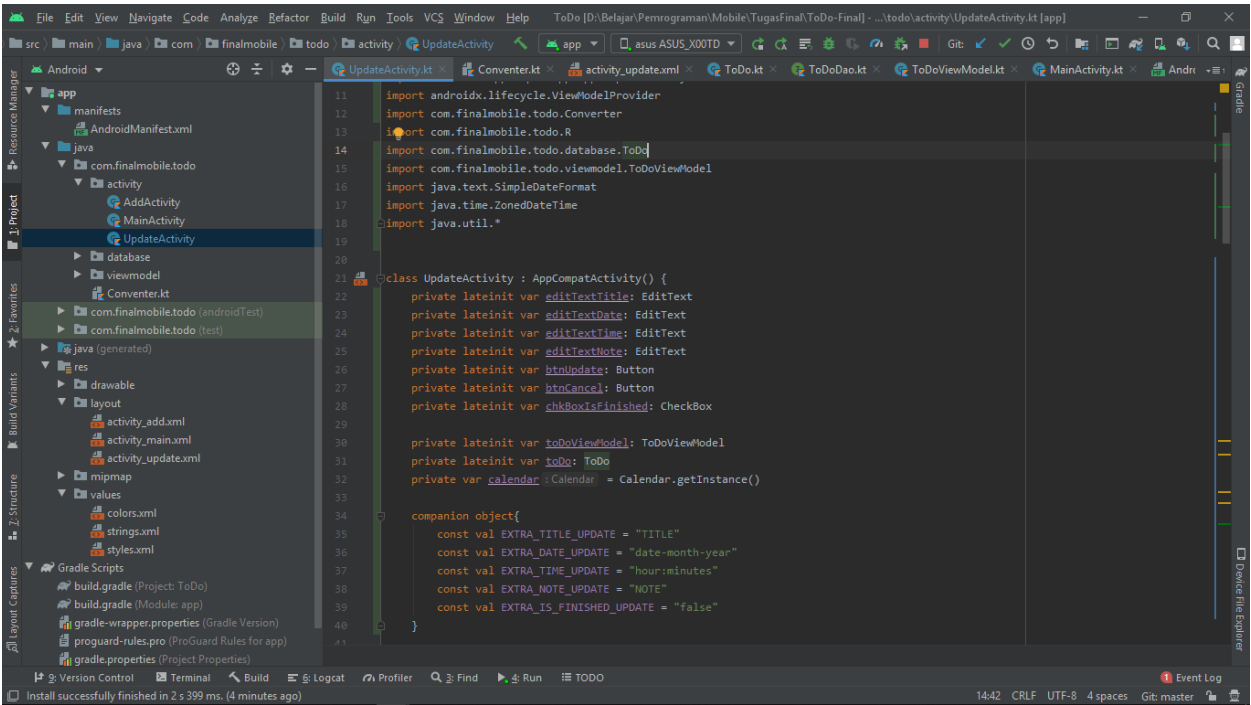
    val current :ZonedDateTime = ZonedDateTime.now(ZoneId.of( zoneId: "+8"))
    val formatter :DateTimeFormatter! = DateTimeFormatter.ofPattern( pattern: "EEE, MMM dd, yyyy, HH:mm:ss")
    val strCreatedDate :String! = current.format(formatter)
    val createdDate :Int = Converter.dateToInt(current)
    val strDueDate :String = editTextDate.text.toString().trim()
    val dueDate :Int = Converter.stringDateToInt(strDueDate)
    val strDueHour :String = editTextTime.text.toString().trim()
    val dueHour :Int = Converter.stringTimeToInt(strDueHour)
    val title :String = editTextTitle.text.toString().trim()
    val note :String = editTextNote.text.toString().trim()

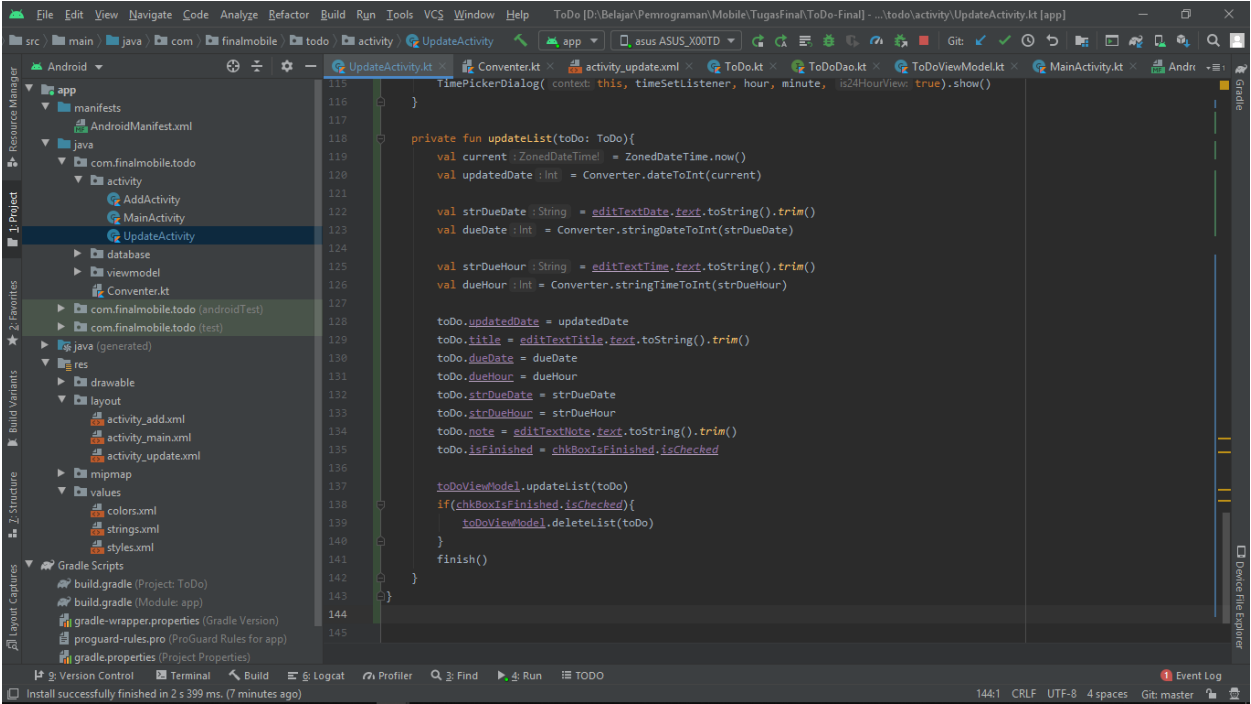
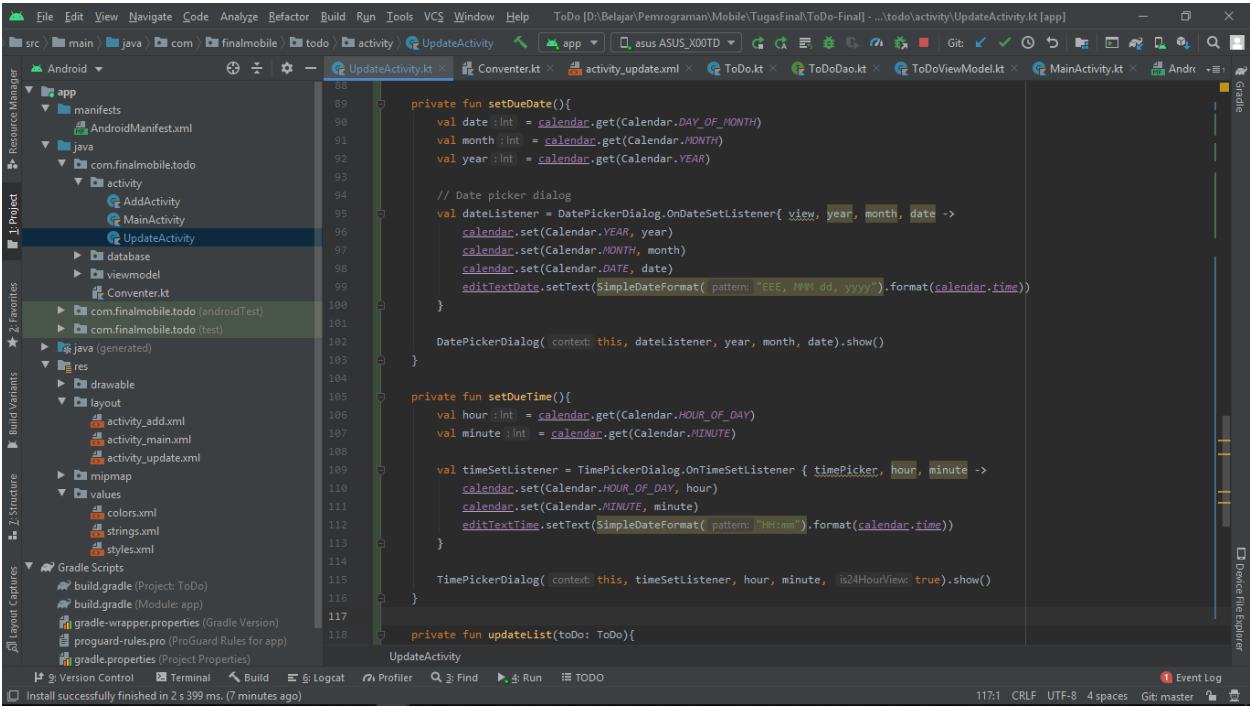
    toDoViewModel.insertList(
        ToDo(
            createdDate = createdDate,
            strCreatedDate = strCreatedDate,
            title = title,
            dueDate = dueDate,
            dueHour = dueHour,
            strDueDate = strDueDate,
            strDueHour = strDueHour,
            note = note,
            isFinished = false
        )
    )
    finish()
}
```

Menambahkan Fitur Toast Untuk Mengatasi Masalah Saat Menyimpan Data Kosong

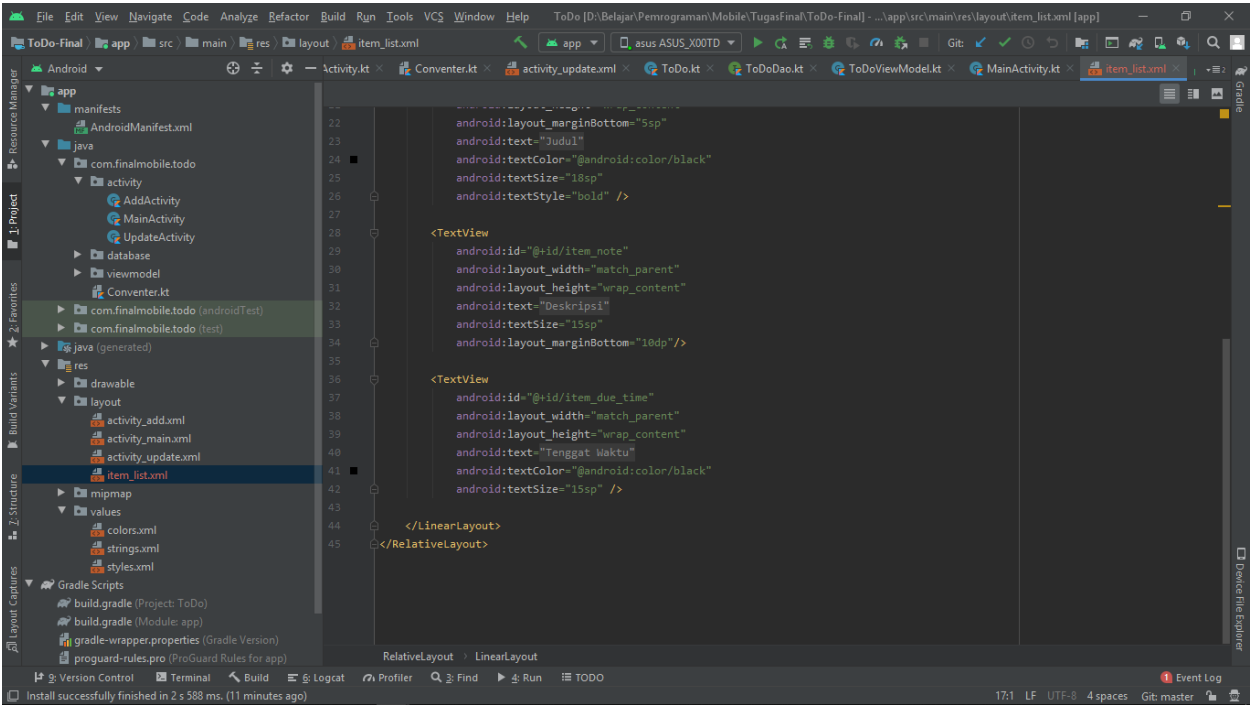
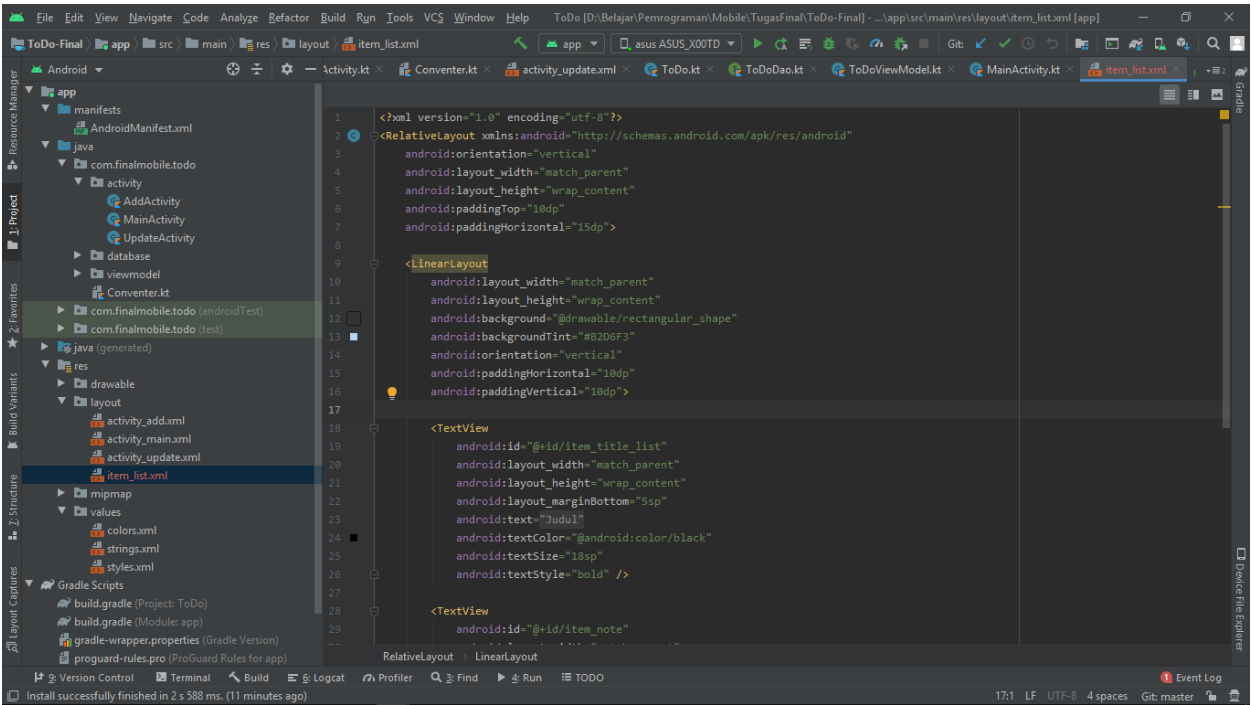


Menambahkan Isi UpdateActivity dan Menyesuaikan dengan Layout

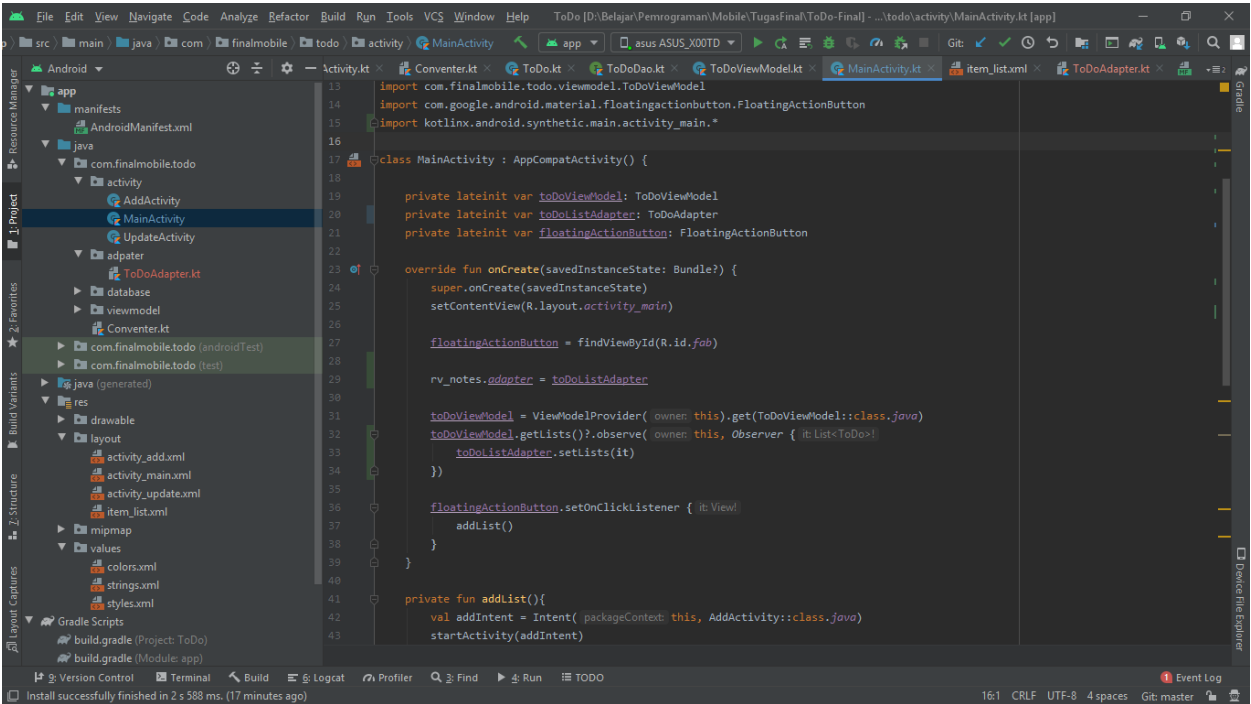
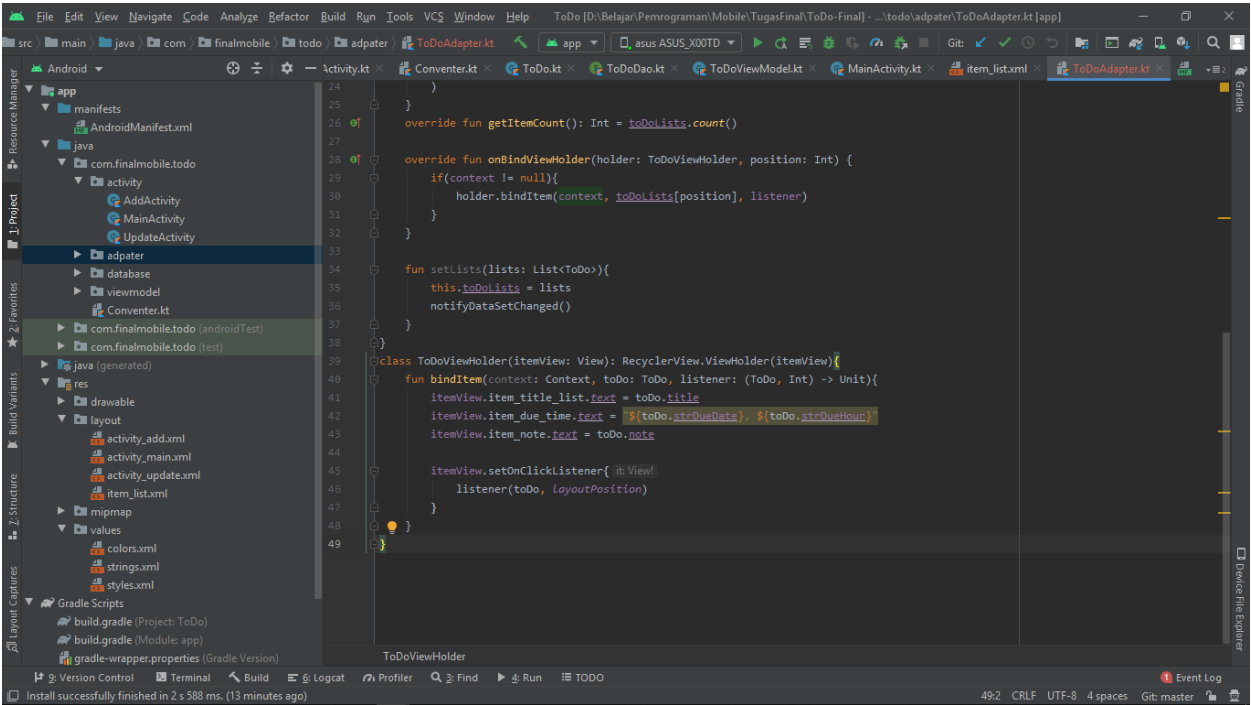
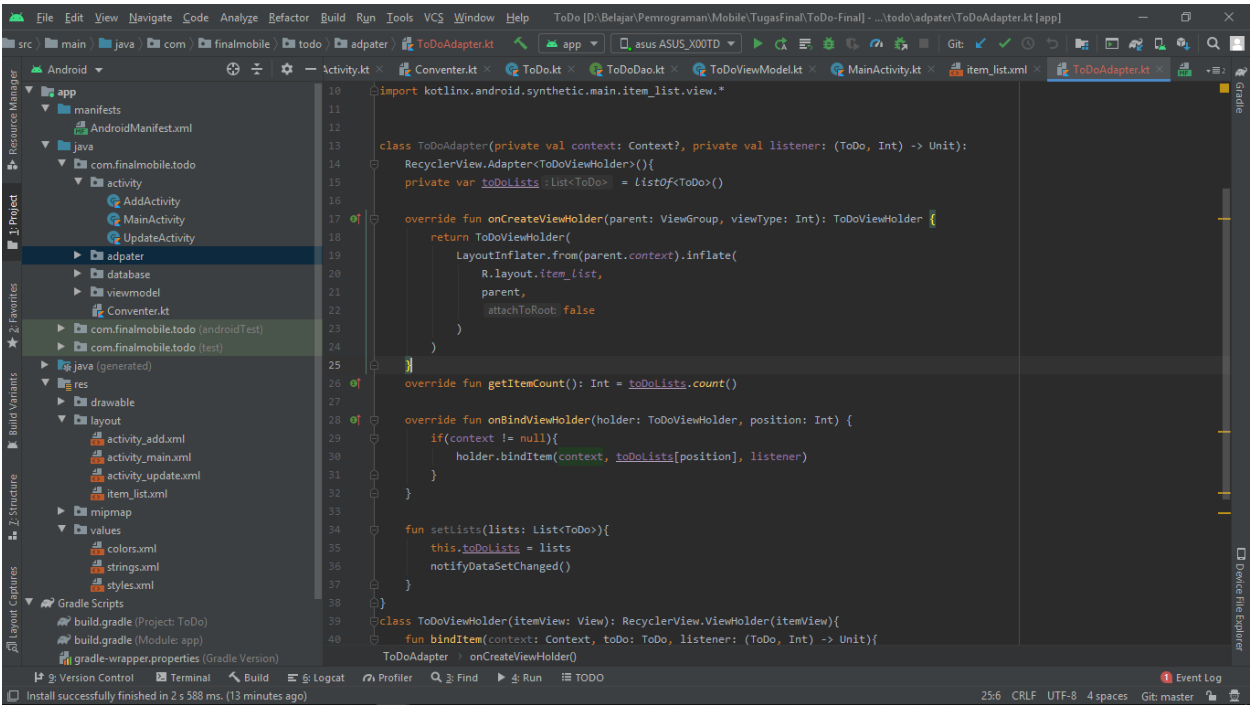




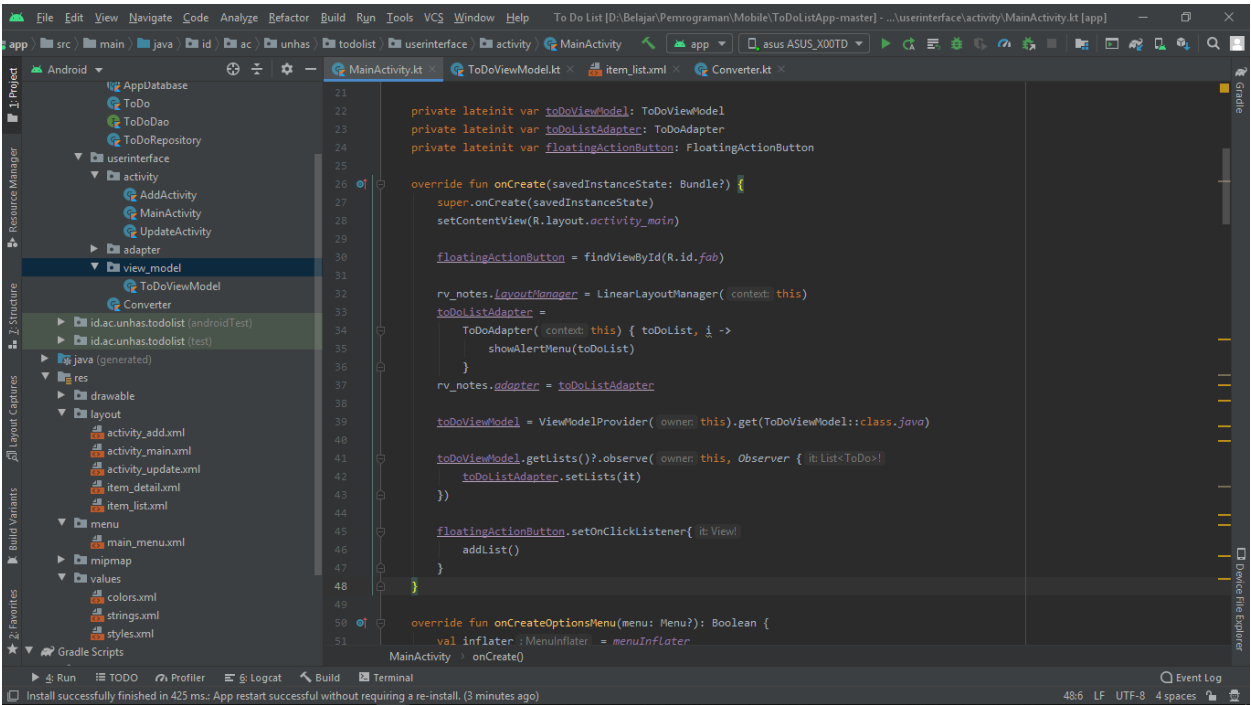
Membuat Layout item_list



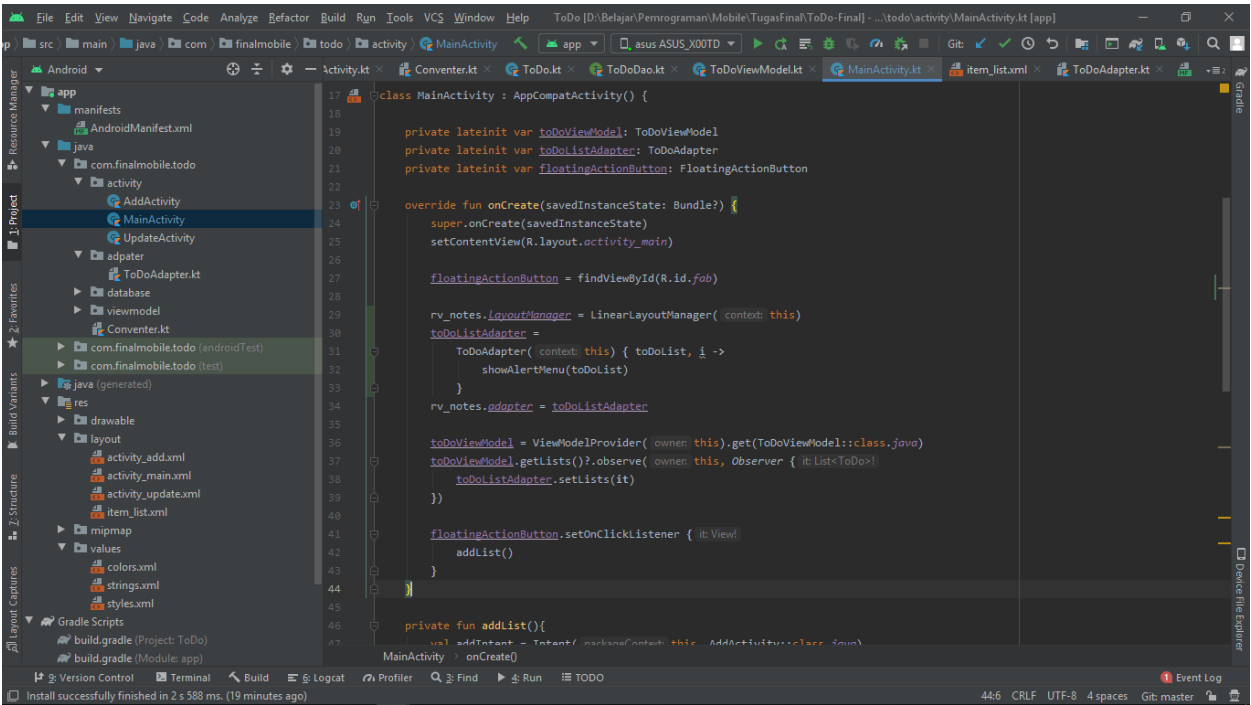
Membuat Adapter



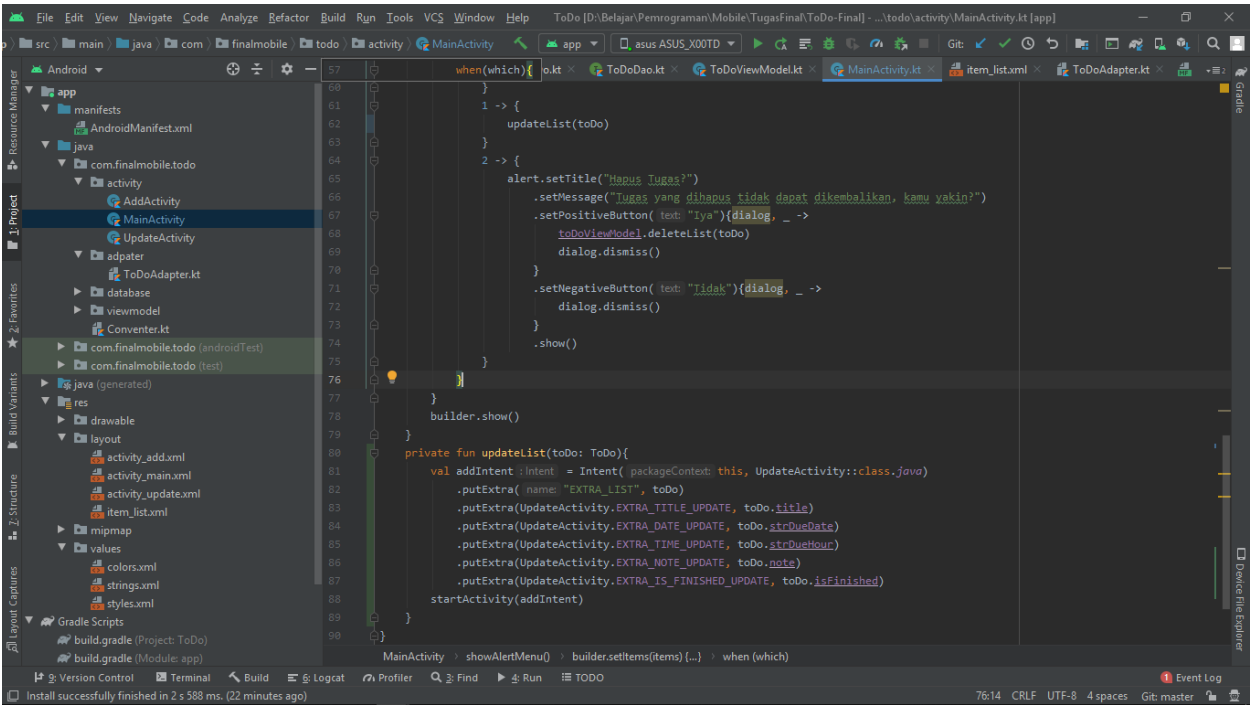
Menghubungkan MainActivity dengan ViewModel



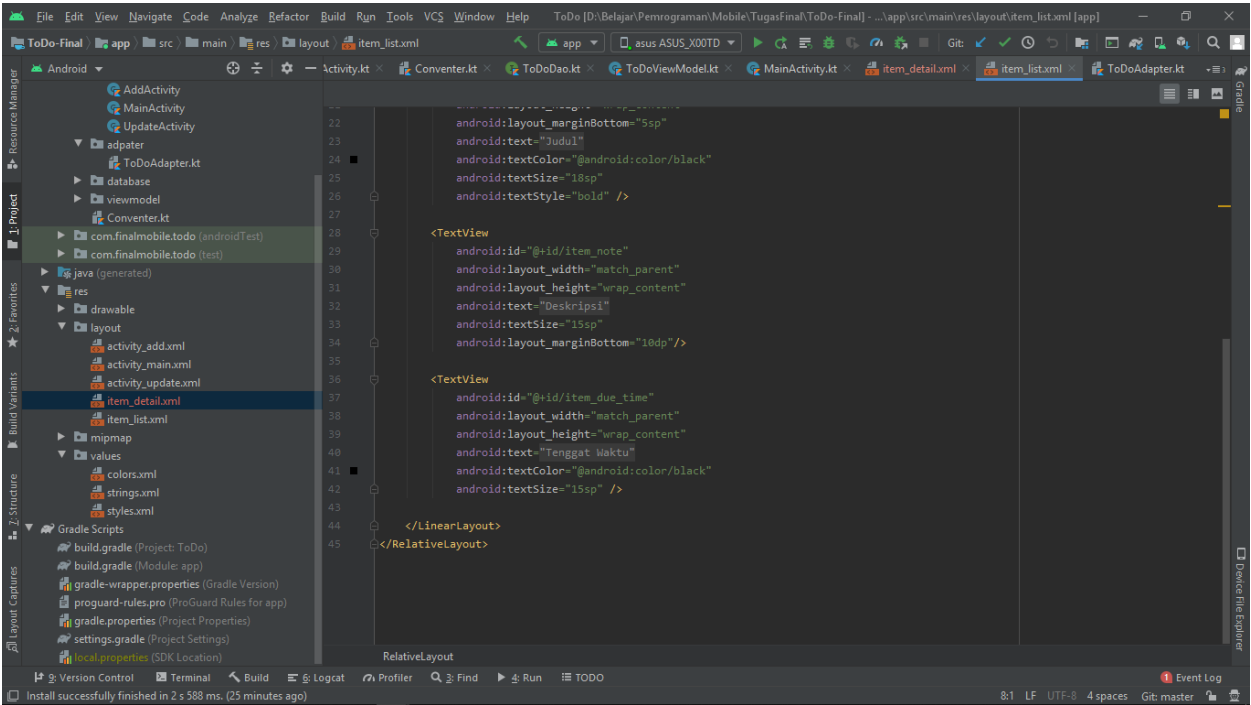
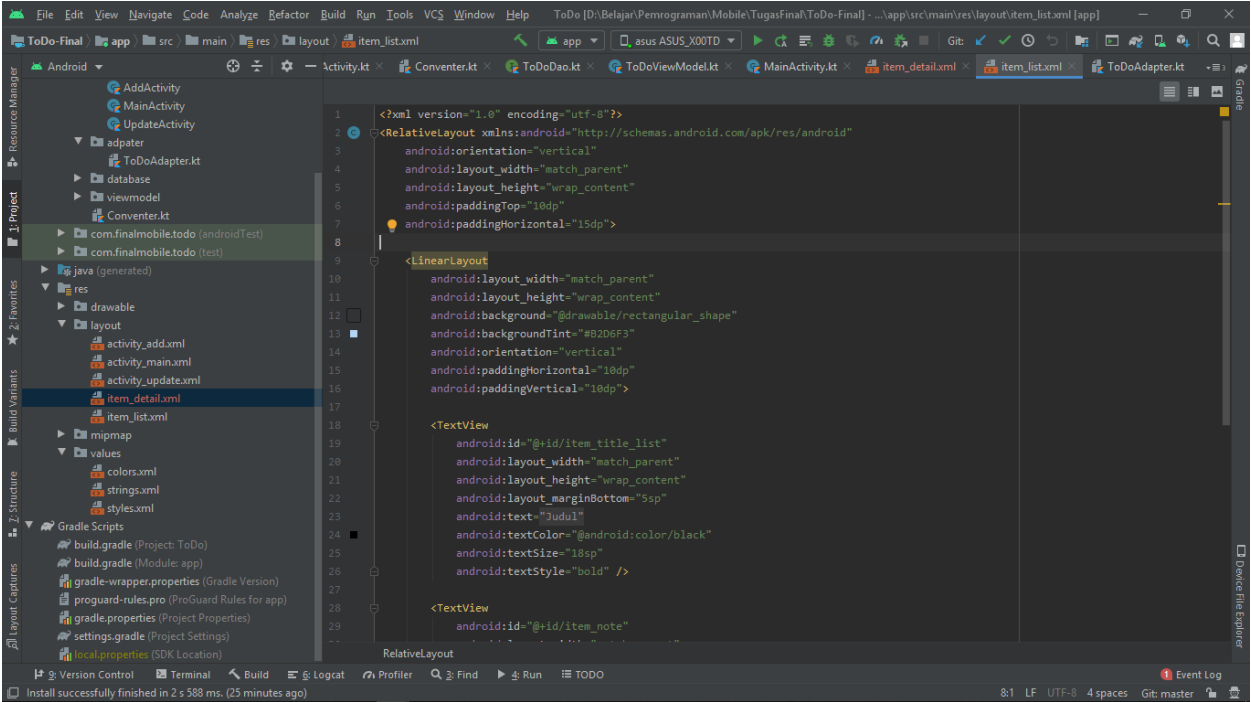
Menghubungkan fitur alert di MainActivity



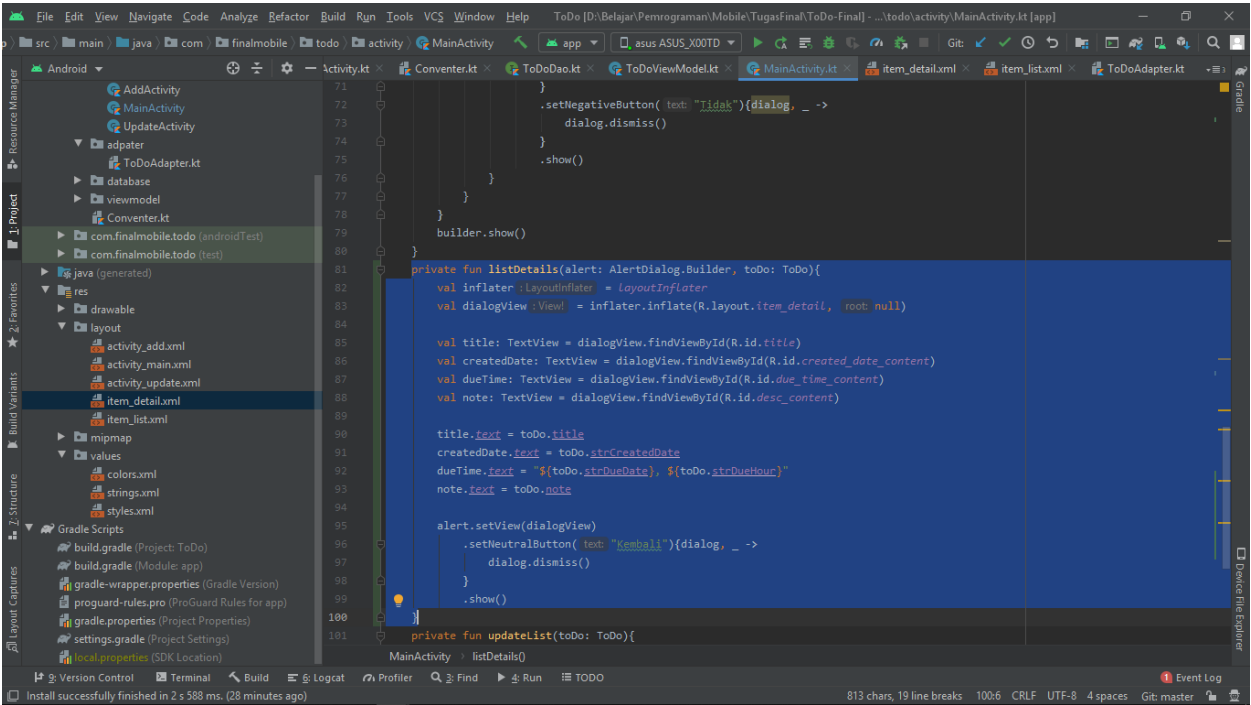
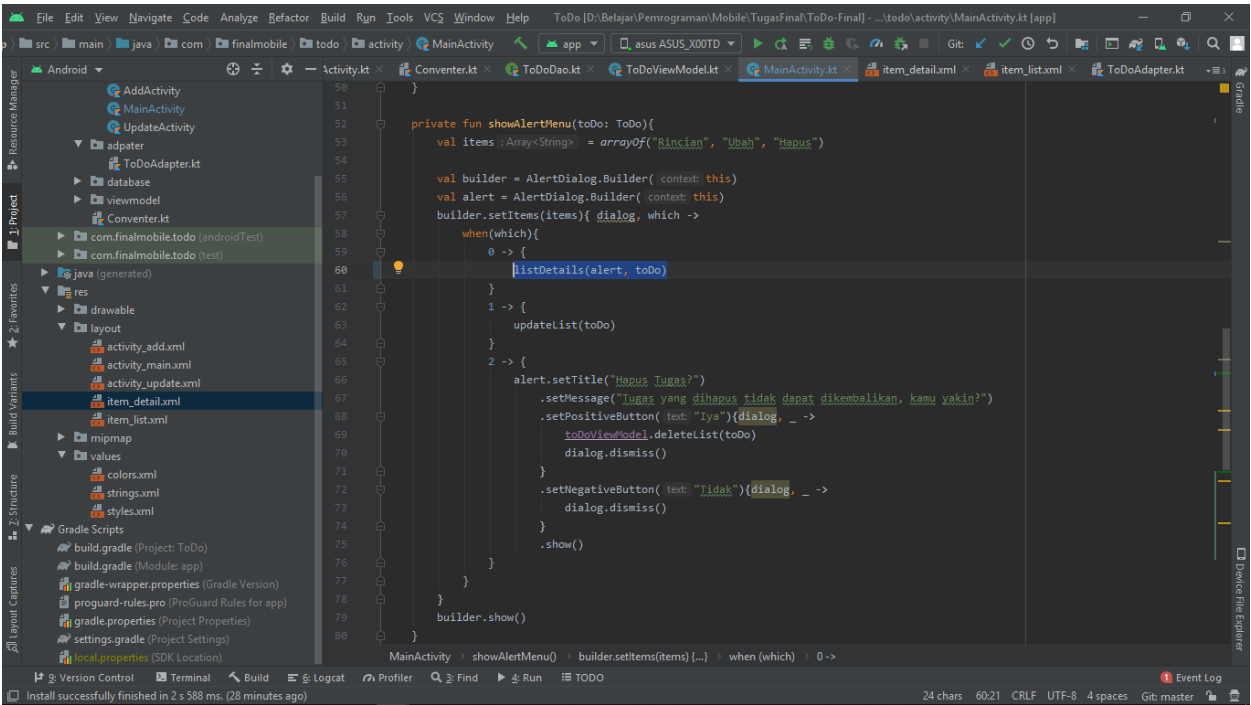
Membuat Fitur Update di Main Activity



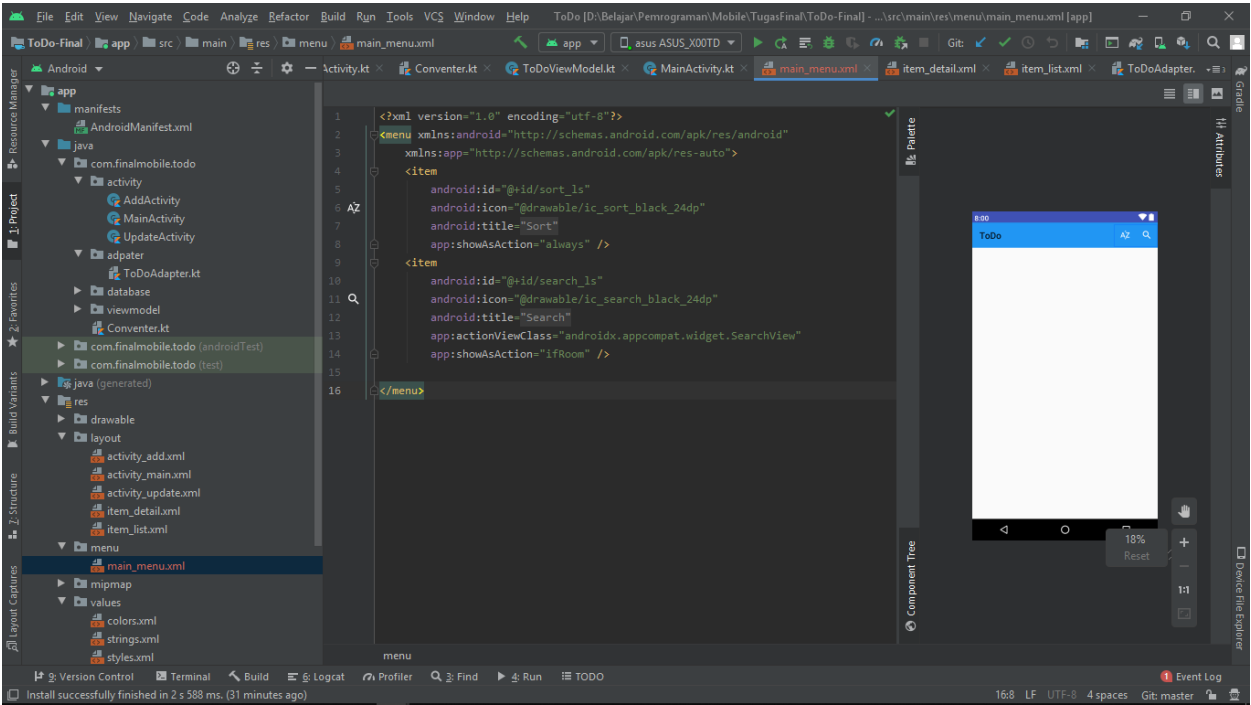
Membuat Layout item_detail



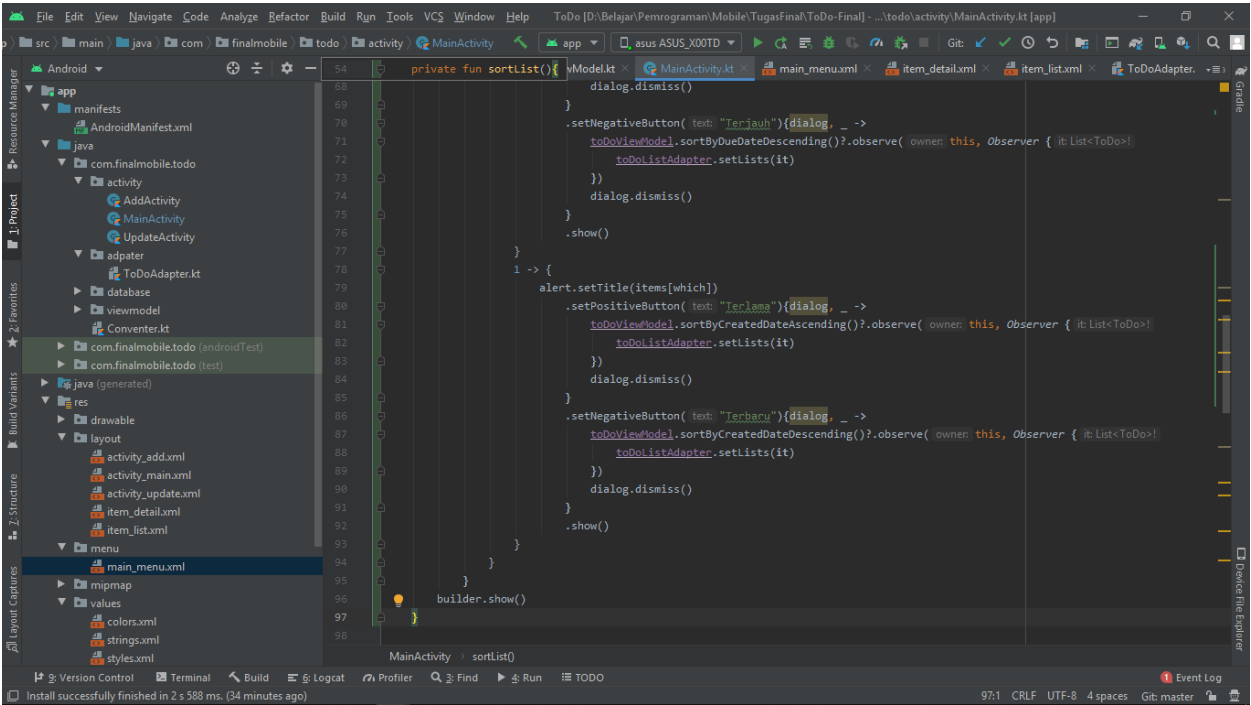
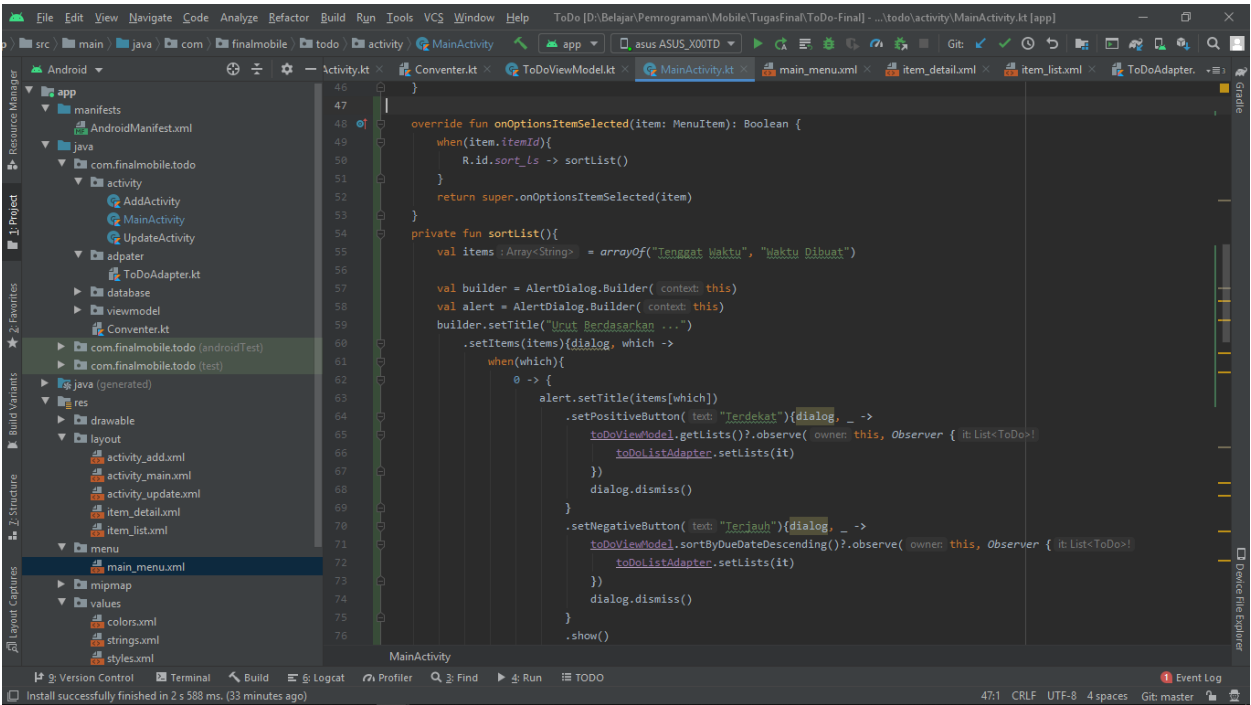
Membuat Fitur Detail di Main Activity



Membuat Layout main_menu dan membuat fitur sort dan search



Membuat Fitur Sort pada Main Menu



Membuat Fitur Search pada MainActivity

